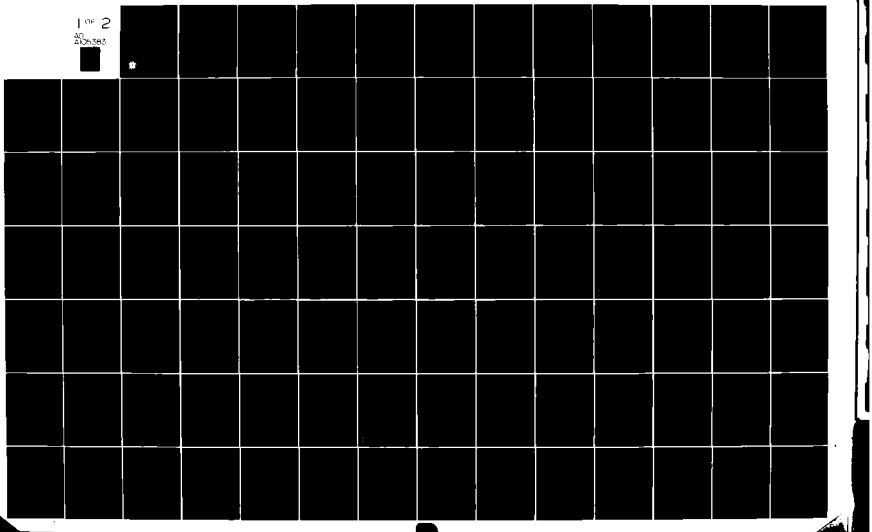
AD-A105 383 NOTRE DAME UNIV IN DEPT OF ELECTRICAL ENGINEERING F/G 9/2
NEW APPROACH TO SPEECH DIGITIZATION COMBINING TIME-DOMAIN HARMO--ETC(U)
AUG 81 J L MELS, A K PANDE DCA100-80-C-0050
UNCLASSIFIED NL



Department of

ELECTRICAL ENGINEERING

UNIVERSITY OF NOTRE DAME, NOTRE DAME, INDIANA

FINAL REPORT

NEW APPROACH TO SPEECH DIGITIZATION COMBINING TIME DOMAIN HARMONIC SCALING AND ADAPTIVE RESIDUAL CODING

Prepared for

Defense Communications Agency
Defense Communications Engineering Center
1860 Wiehle Avenue
Reston, Virginia 22090

Contract No. DCA 100-80-C-0050

31 August 1981

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO. AD-A105383

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)

A NEW APPROACH TO [illegible]L SPEECH DIGITIZATION COMBINING TIME-DOMAIN HARMONIC SCALING AND ADAPTIVE RESIDUAL CODING.

5. TYPE OF REPORT & PERIOD COVERED

Final Report, Oct [illegible]980 - Aug[illegible]981

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)

James L. Melsa and Arun K. Pande

8. CONTRACT OR GRANT NUMBER(s)

DCA 1[illegible]-8Ø-C-ØØ5Ø

9. PERFORMING ORGANIZATION NAME AND ADDRESS

Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS

Defense Communications Agency
Contract Management Division, Code 260
Washington, D.C. 20305

12. REPORT DATE

Aug[illegible]981

13. NUMBER OF PAGES

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

174

15. SECURITY CLASS. (of this report)

Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce, for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Speech coding, 9600 bps speech transmission, 16000 bps speech transmission, pitch extraction, adaptive residual coder, waveform reconstruction, time-domain harmonic scaling.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the results of a twelve-month study, supported under DCA Contract 100-80-C-0050, of a new speech digitization algorithm combining Time-Domain Harmonic Scaling (TDHS) and Adaptive Residual Coding (ARC). The FORTRAN simulation of this system conducted as part of this study produces high quality speech reproduction at mediumband bit rates of 9.6 kb/s and 16 kb/s. This system also displays excellent robustness characteristics for channel bit error rates as high as 1% and for acoustic background noise. By basing the (over)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified 388467

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

required pitch extraction on a three-level clipped signal, the hardware requirements for the system are kept modest.

The combined algorithm has several features which become significant in a full system application. Because the algorithm is a high performance, waveform matching algorithm, extremely good performance in the randem configuration with other algorithms is anticipated. Since the technique is basically a waveform reconstruction technique, it will perform well on non speech signals such as in-band signaling and modem tones.

ABSTRACT

This report describes the results of a twelve-month study, supported under DCA Contract 100-80-C-0050, of a new speech digitization algorithm combining Time-Domain Harmonic Scaling (TDHS) and Adaptive Residual Coding (ARC). The FORTRAN simulation of this system conducted as part of this study produces high quality speech reproduction at medium band bit rates of 9.6 kb/s and 16 kb/s. This system also displays excellent robustness characteristics for channel bit error rates as high as 1% and for acoustic background noise. By basing the required pitch extraction on a three-level clipped signal, the hardware requirements for the system are kept modest.

The combined algorithm has several features which become significant in a full system application. Because the algorithm is a high performance, waveform matching algorithm, extremely good performance in the tandem configuration with other algorithms is anticipated. Since the technique is basically a waveform reconstruction technique, it will perform well on non speech signals such as in-band signaling and modem tones.

PROJECT PERSONNEL

James L. Melsa, principal investigator

Arun K. Pande, research assistant

FINAL REPORT

DCA Contract 100-80-C-0050

CHAPTER 1

INTRODUCTION AND OUTLINE OF REPORT

1.1 INTRODUCTION

This report describes the results of a twelve-month study supported under DCA Contract 100-80-C-0050 of a new speech digitization algorithm combining Time-Domain Harmonic Scaling (TDHS) and Adaptive Residual Coding (ARC). The FORTRAN simulation of this system conducted as part of this study produces high quality speech reproduction at medium band bit rates of 9.6 kb/s and 16 kb/s. This system also displays excellent robustness characteristics for channel bit error rates as high as 1% and for acoustic background noise. By basing the required pitch extraction on a three-level clipped signal, the hardware requirements for the system are modest.

The TDHS algorithm was developed by Malah (1979) and applied by him (Malah, 1980) to a CVSD system at a transmission rate of 7.2 kb/s. More recently Malah (1981) has combined TDHS with Transform Coding and Sub-band Coding at mediumband bit rates. This algorithm consists of properly weighting several adjacent input signal segments of pitch dependent duration by suitable window functions. As a result of this, the number of samples to be transmitted can be reduced by a factor of two. If the bit rate is kept the same, the number of bits allowed per sample is doubled, and the performance of the coder can be improved significantly. The ARC structure was developed by Cohn and Melsa (1975b) and implemented in hardware by CODEX (Qureshi and Forney, 1975). This structure involves the combination of pitch compensating adaptive quantizer (Cohn and Melsa,

1976), sequentially adaptive linear predictor, and adaptive source coding.

The combined algorithm has several features which become significant in a full system application. Because the algorithm is a high performance, waveform matching algorithm, extremely good performance in the tandem configuration with other algorithms is anticipated. Since the technique is basically a waveform reconstruction technique, it will perform well on non speech signals such as in-band signaling and modem tones.

1.2 ALGORITHM OBJECTIVE

The following objective for the speech coding algorithm have been established from the Statement of Work:

1. The speech processing system shall operate at a medium band transmission data rates of 9.6 kb/s and 16 kb/s.
2. The speech processing system shall produce toll quality speech reproduction.
3. The audio bandwidth of the input speech shall be greater than or equal to 3200 Hz.
4. The speech coder shall produce good quality speech under conditions of a random transmission bit error rate of 1 percent.
5. The speech coder shall produce toll quality speech under 60 dB (reference to 20 μ newtons/meter^2) of acoustic background noise such as office noise and good quality speech under 100 dB of acoustic background noise.
6. The computational complexity of the algorithm shall be minimized.
7. The system shall be capable of processing non-speech signals, such as in-band signaling and modem tones.
8. Tandem connection with other algorithms such as CVSD and LPC-10 should cause negligible distortion.

1.3 OUTLINE OF THE REPORT

The design and development of TDHS-ARC algorithm is described in three chapters. Chapter 2 contains the research work pertaining to Time Domain Harmonic Scaling. The recent development of TDHS is described to provide the necessary background material. The research problems such as sampling rate, window design, compression ratio and pitch extraction are addressed in this chapter. The design of an Adaptive Residual Coder for the

frequency compressed speech signal is outlined in Chapter 3. Various objective performance measure criteria and a new fixed wordlength source code are also described in this chapter. The complete system structure is presented in Chapter 4. The effect of transmission errors and background noise on the system performance is given. The strategy to control the buffer behaviour and the scheme of extracting pitch at the receiver are also discussed in this chapter. Chapter 5 describes the modifications of the algorithm which are needed to operate at 16 kb/s.

The Fourier Transform of several TDHS window functions are given in Appendix A. The flow charts for the FORTRAN simulation are given in Appendix B while the source listings are given in Appendix C.

CHAPTER 2

TIME DOMAIN HARMONIC SCALING

2.1 INTRODUCTION

In Chapter 1, it was indicated that the system uses time domain harmonic scaling (TDHS) to reduce the number of speech samples to be transmitted without causing excessive distortion. This process also allows an increased number of bits per sample to be available for coding. The TDHS algorithm uses a pitch adaptive window to perform frequency compression or expansion on the speech signal. Such frequency scaling operations are dependent on various factors such as the type of the window and the pitch extraction method used and the amount of frequency compression employed. In this chapter, these and other factors affecting TDHS performance are discussed and results are presented.

2.2 FREQUENCY COMPRESSION AND EXPANSION

The time varying Fourier representation of speech has successfully been used in vocoders. The techniques used for frequency scaling in these vocoders are fairly complex and, therefore, have not been extended to mediumband speech coders. Recently, a time-domain algorithm for frequency scaling was developed by Malah [April, 1979] and applied by him [April, 1980] to CVSD system at a transmission rate of 7200 bits/ second.

The algorithm is quite general and involves choices of such parameters as windowing function and scaling factors. One specific, and most common, form of the the algorithm is presented below using triangular

windows and 2 to 1 scaling. The TDHS algorithm makes use of the long-term pitch redundancy of speech signals in a manner that is similar to gapped analysis [Melsa, et. al., 1980]. However, by a clever choice of the time-domain windowing function, the TDHS algorithm is able to ensure continuity across the frame boundaries. At the transmitter, the basic concept is to compress two pitch periods of speech into a single pitch period of the same time duration but at half the sampling rate. At the receiver, the compressed signal is frequency multiplied to reconstitute an approximation of the original input signal. Consider first the frequency compression operation.

Suppose that speech samples up to sample number k_0 have been processed; the corresponding output sample number is $m_0 = k_0/2$. The first step is to determine the pitch period associated with the samples following k_0 by any standard method such as correlation or AMDF. Let the resulting pitch period, in samples, be N_p. The value of N_p during unvoiced speech or silence can be set arbitrarily. Consider the $2N_p$ samples from k_0+1 to k_0+2N_p as shown in Fig. 2.1. Note that these samples need not be pitch synchronous. These $2N_p$ samples are frequency compressed into N_p samples by use of the following TDHS algorithm where y(m) is the compressed output and s(k) is the original speech.

$$y(m_0+i) = s(k_0+i)\, h(i:N_p) + s(k_0+N_p+i)\,[1 - h(i:N_p)] \qquad i = 1,2,\ldots,N_p \qquad (2.1)$$

Here $h(i:N_p) = \begin{cases} 1 - (i-1)/(N_p-1) & 1 \leq i \leq N_p \\ 0 & \text{otherwise} \end{cases}$

Equation (2.1) can be rewritten as

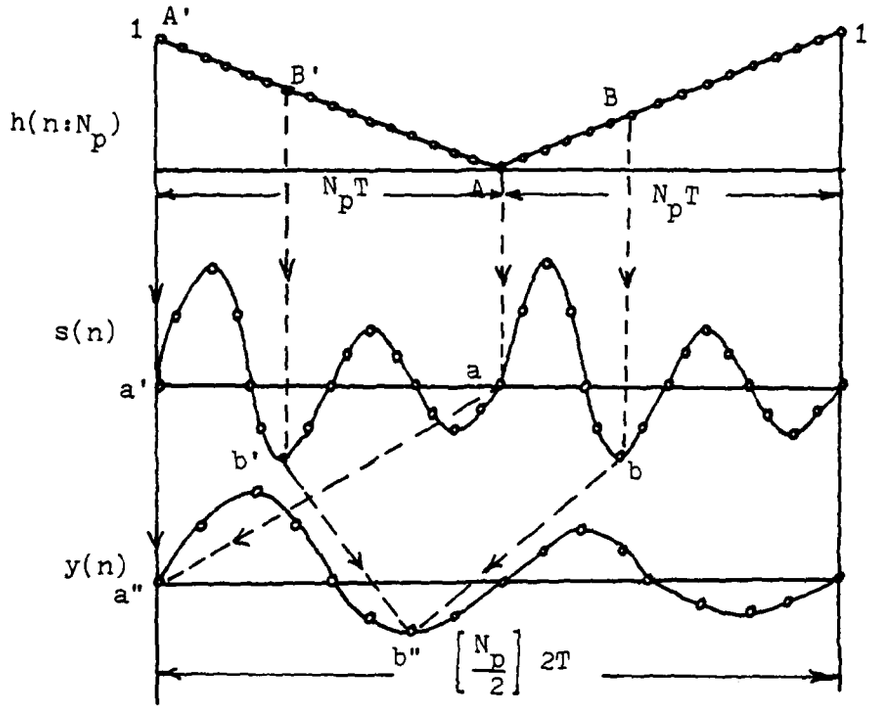


Fig. 2.1 TDHS frequency compression for the compression ratio of 2:1.

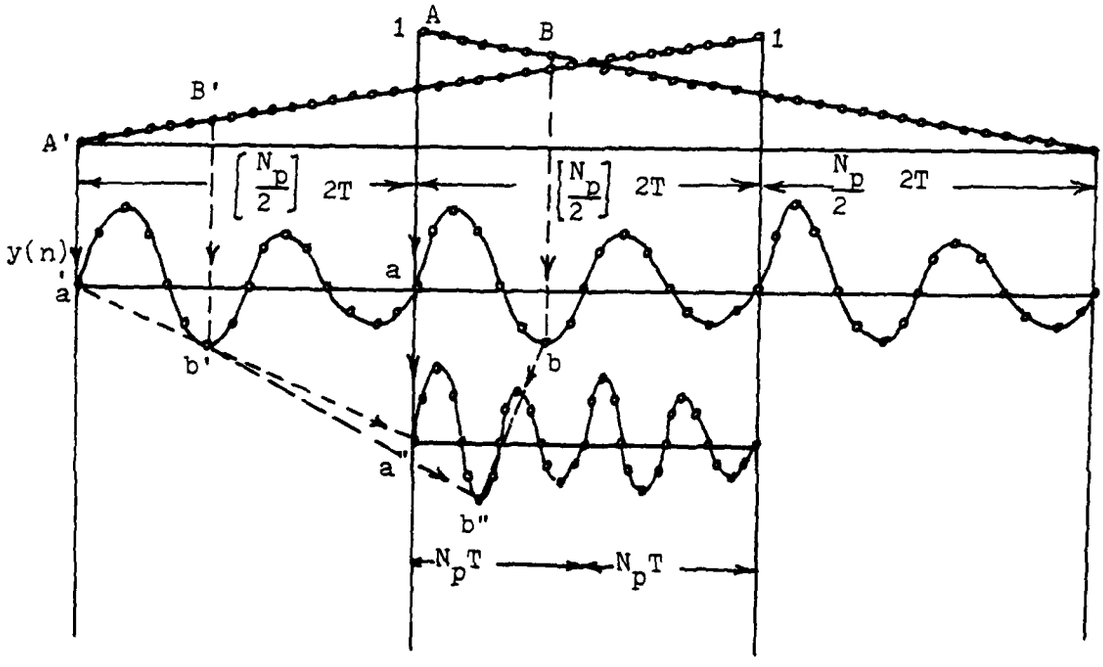


Fig. 2.2 TDHS frequency expansion for the expansion ratio of 1:2.

$$y(m_0+i) = s(k_0+N_p+i) + h(i:N_p)[s(k_0+i) - s(k_0+N_p+i)] \quad (2.2)$$

to indicate that only one multiplication and two additions are required per output sample. The frequency compression operation is illustrated in Fig. 2.1.

As long as the window function $h(i:N_p)$ satisfies the properties

$$\begin{aligned} h(1:N_p) &= 1 \\ h(N_p:N_p) &= 0 \end{aligned} \quad (2.3)$$

the following continuity conditions will be satisfied

$$\begin{aligned} y(m_0) &= s(k_0) \\ y(m_0+1) &= s(k_0+1) \end{aligned} \quad (2.4)$$

and

$$\begin{aligned} y(m_0+N_p) &= s(k_0+2N_p) \\ y(m_0+N_p+1) &= s(k_0+2N_p+1) \end{aligned} \quad (2.5)$$

At the receiver, it is necessary to use a frequency multiplication procedure to regenerate the $2N_p$ samples from the N_p samples of $y(m)$. Using the TDHS algorithm this is accomplished as

$$\hat{s}(k_0+i) = y(m_0+i)\, h(i:2N_p) + y(m_0-N_p+i)[1 - h(i:2N_p)] \quad i = 1,2,3,\ldots,2N_p \quad (2.6)$$

The frequency multiplication operation is illustrated in Fig. 2.2. Once again if the window function satisfies Eq. (2.3), continuity will be ensured across the frame boundary.

The frequency spectrum of the original speech, compressed speech and expanded speech is shown in Figs. 2.3 & 2.4. The plot is for 20 msec of voiced speech. It can be seen that frequency spectrum of original and expanded speech match very well.

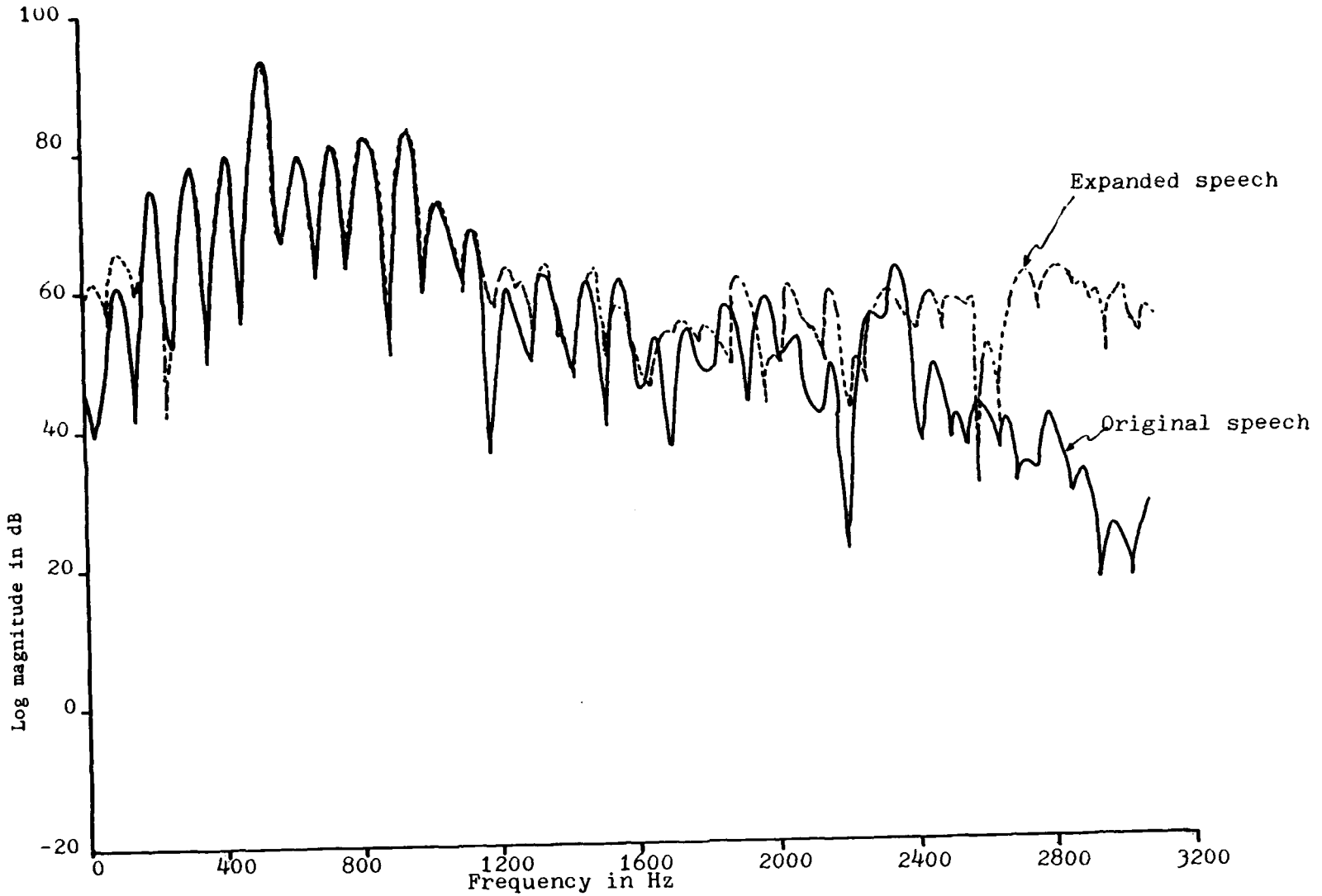


Fig. 2.3 Frequency spectrum of voiced segment of original and expanded speech.

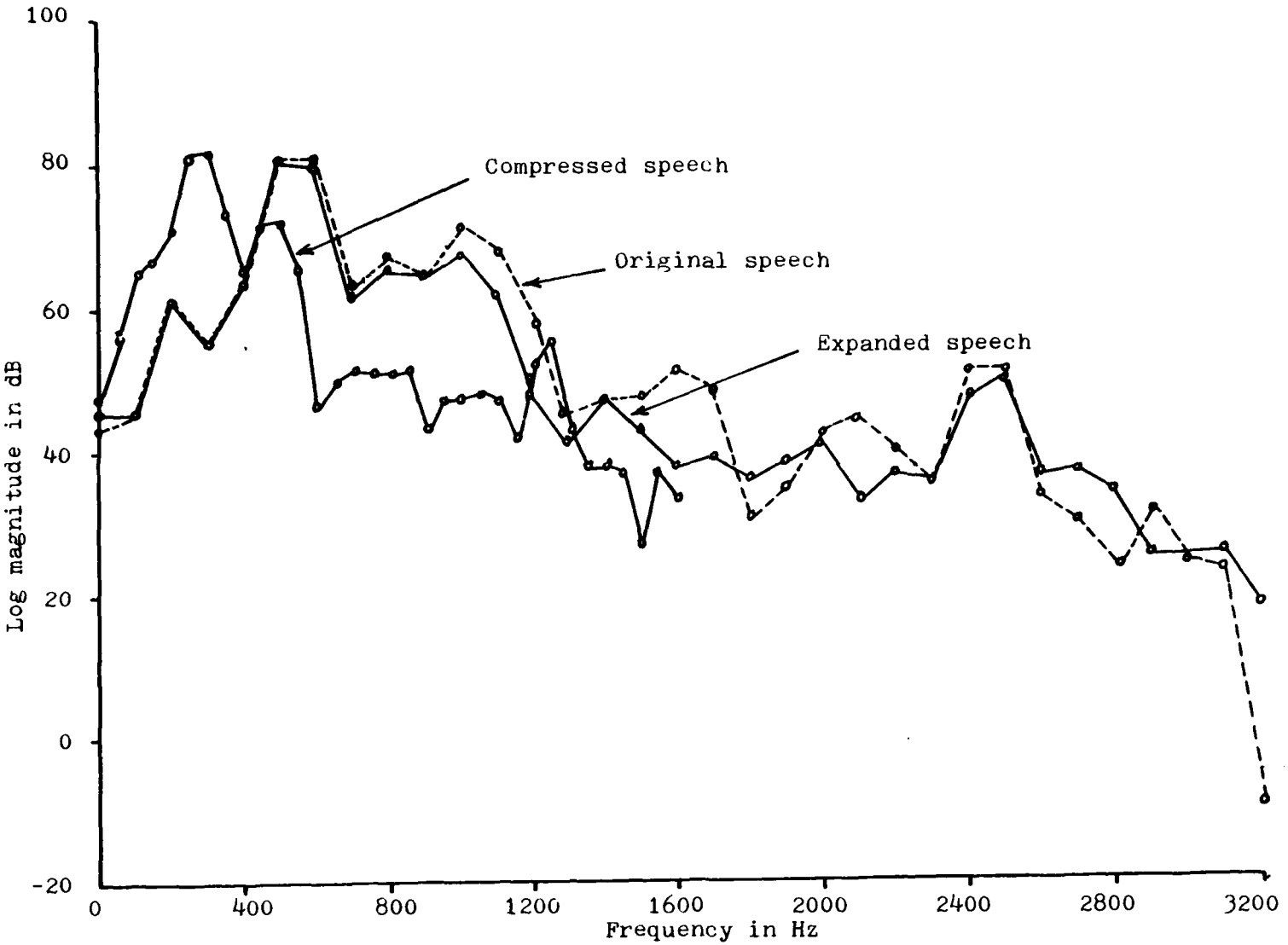


Fig. 2.4 Frequency Spectrum

2.3 SAMPLING RATE

Samples of analog signals are a unique representation if the analog signal is bandlimited and if the sampling rate is more than twice the Nyquist frequency. Speech signals are not inherently bandlimited, although the spectrum does fall off rapidly at high frequencies. For voiced sounds, the high frequencies are more than 40 db below the peak of the spectrum for frequencies above 4 kHz. On the other hand for unvoiced sounds, the spectrum does not fall off appreciably even above 8 kHz. However, telephone transmission has a bandlimiting effect on speech signals and the maximum frequency in speech signals can be considered as 3.2-3.5 kHz for conversational or "telephone quality" speech.

TDHS algorithms are based on the assumption that the fundamental frequency F_o (the pitch) of the input voiced-speech signal is known. If estimated pitch frequency is F_p then error in frequency estimate is $F_p - F_o$. This error in pitch estimation can be tolerated [Malah, 1979] if

$$\frac{|F_p - F_o|}{F_p} < \frac{1}{2L}$$

where L = number of harmonics present in bandlimited periodic input signal. It is obvious that accuracy in the determination of the pitch period is important. Since a pitch estimator extracts pitch in terms of integer number of samples, the accuracy of the pitch extracted depends on the sampling frequency of the input speech signal. As the sampling frequency is increased, the pitch period resolution is improved. However, by increasing the sampling rate or oversampling the speech signal, fewer bits per sample are available for coding the quantizer levels. For example, if the

sampling rate is 6400 samples/sec (3200 samples/sec for compressed speech) and the transmission rate is 9600 bits/sec, the average number of bits per samples is 3. For a sampling rate of 10000 samples/sec, the pitch period estimation becomes 36% more accurate while the average entropy allowed drops down from 3 bits to 1.92 bits/sample. Hence, there is a trade off between the improvement in performance due to increased pitch accuracy and the degradation due to the decrease in entropy.

To study this trade-off, input speech was sampled at 3 different sampling frequencies, namely: 6.4, 8 and 9.6 kHz. First only frequency compression and expansion operations were considered. Informal listening tests have shown that unvoiced (higher frequency) speech sounds much better for higher sampling rates than lower ones. However, overall speech quality does not differ significantly. When quantization was introduced (TDHS-ARC System), no significant change in quality was noticed for the different sampling rates. As indicated earlier, fewer bits per sample are available for the higher sampling rates. This results in more quantization noise which masks the improvement obtained in the unvoiced sound by higher sampling rates. The sampling frequency for the system with transmission rate of 9.6 kb/s was chosen to be 6400 Hz. With more bits per sample available for coding in 16 kb/s System, a sampling frequency of 8 kHz may be a good choice.

2.4 WINDOW DESIGN

To determine the proper window function to be used, the requirements and the constraints that should be satisfied by this function need to be discussed.

As mentioned earlier, the TDHS algorithm consists of properly weighing several adjacent input signal segments (with pitch dependent duration) by a suitable window function to produce an output segment. Since the pitch period N_p varies, it is necessary that adjacent segments processed with different values of N_p should maintain output signal continuity at the interface between segments. This could be written in equation form as

$$y(m_0) = s(k_0)$$
$$y(m_0+1) = s(k_0+1) \tag{2.7}$$

and

$$y(m_0+N_p) = s(k_0+2N_p)$$
$$y(m_0+N_p+1) = s(k_0+2N_p+1) \tag{2.8}$$

where k_0 is the sample number up to which speech samples are processed

$m_0=k_0/2$ corresponding output sample number

From Eq. (2.1) it is known that

$$y(m_0+i) = s(k_0+i)\, h(i:N_p) + s(k_0+N_p+i)[1 - h(i:N_p)]$$
$$i = 1,2,\ldots,N_p$$

for $i = 1$

$$y(m_0+1) = s(k_0+1)\, h(1:N_p) + s(k_0+N_p+1)[1 - h(1:N_p)] \tag{2.9}$$

for $i = N_p$

$$y(m_0+N_p) = s(k_0+N_p)\, h(N_p:N_p) + s(k_0+2N_p)[1 - h(N_p:N_p)] \tag{2.10}$$

To satisfy the continuity conditions in Eqs. (2.7) and (2.8),

$$h(1:N_p) = 1$$

and

$$h(N_p:N_p) = 0$$

Another constraint is imposed by the fact that if this algorithm is used for periodic signals, then exact frequency scaling should be obtained. If the signal has period N_p and $h(n:N_p)$ is the window function,

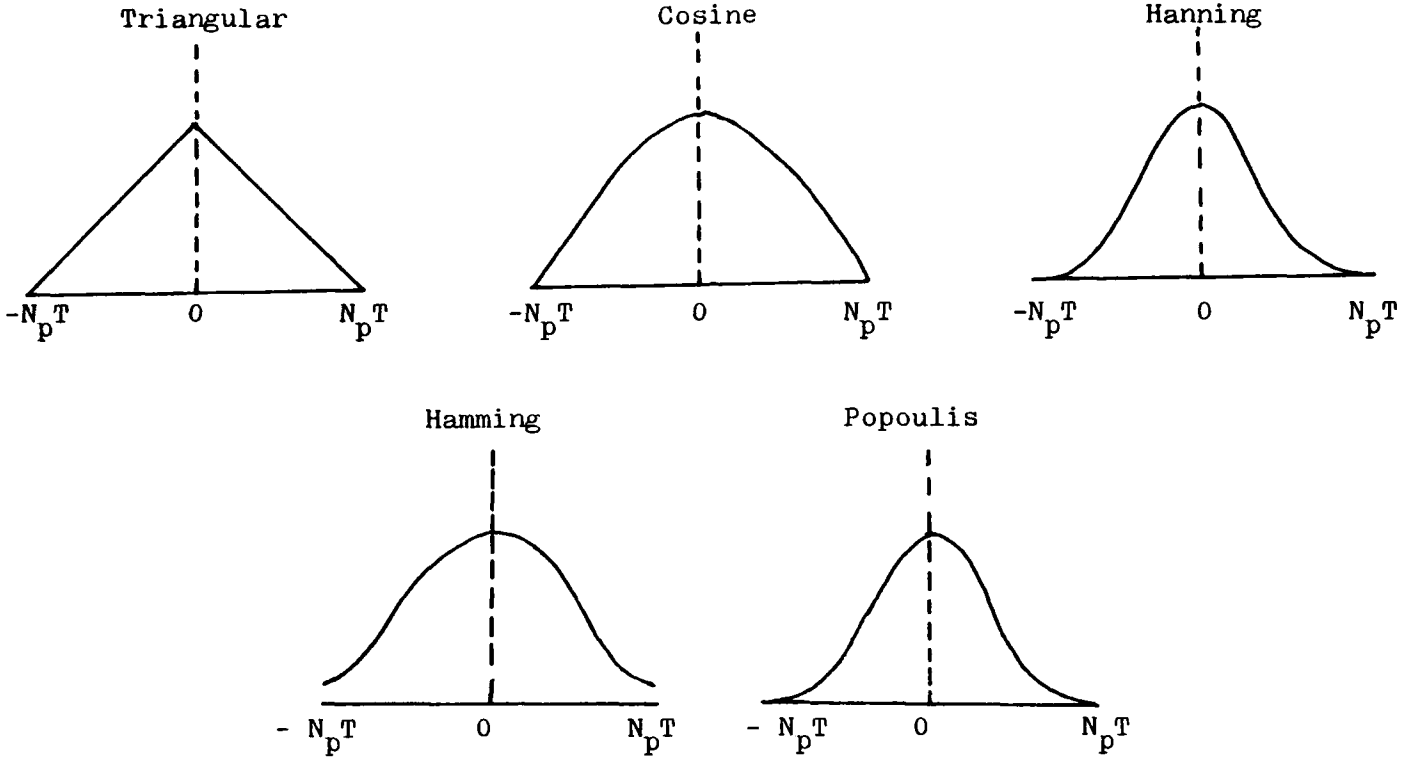


Fig. 2.5 Different window functions.

frequency division by two can be achieved as follows

$$y(n) = s(n)\, h(n:N_p) + s(n+N_p)\, h(n+N_p:N_p)$$
$$n = 1,2,3,\ldots,N_p$$

Since s(n) is periodic

$$s(n) = s(n+N_p)$$

hence, $$y(n) = s(n)[h(n:N_p) + h(n+N_p:N_p)].$$

For exact frequency division by two, it is therefore necessary that $h(n:N_p)$ satisfy

$$h(n:N_p) + h(n+N_p:N_p) = 1 \qquad (2.11)$$

or

$$h(n+N_p:N_p) = 1 - h(n:N_p)$$

There are various types of window functions possible which satisfy above constraints and hence could be used in TDHS algorithm. Some of the possible window functions are shown in Fig. 2.5.

The choice of window depends upon the simplicity of implementation, the number of computation required and the performance. The performance of a particular window is measured in terms of the quality of output speech produced with that window choice. The best method of measuring the quality of output speech is by listening to it. However, this criterion is subjective and besides, very time consuming to use. The other criteria which are frequently used for measuring the quality of speech are segmental signal-to-noise ratio (SEGSNR) in the time and frequency domain. The SEGSNR in the time domain is the average of SNRs calculated for all the segments of speech. The typical length of the segment is 20 msec. The SEGSNR in the frequency domain is calculated in a similar way except the SNRs are computed for the frequency components of speech samples. The frequency components are obtained by taking a DFT of the input and output speech segments.

It was found that SEGSNR in the frequency domain very closely reflects the quality of output speech and thus, could form a good objective measure for this system. The details are discussed in Chapter 3. Table 2.1 shows various window functions and their performance. The frequency response of these functions are outlined in Appendix A. It can be seen from Table 2.1 that the performance of the TDHS algorithm for different window functions and obtained for a two second male utterance is almost the same. However, the complexity of these windows vary considerably. For example, the triangular window is very simple to implement while its performance is slightly worse than Hanning window which requires more computations.

Figure 2.6 shows the TDHS output speech plot for the word, "CATS" for different types of window functions. For the trapezoidal and Tukey windows the energy fluctuations in transition regions are more accurately reconstructed than the rest. The trapezoidal window function could be an attractive alternative to triangular window function since further savings in multiplication operations could be achieved. This is demonstrated as follows. From Table 2.1, the trapezoidal window function is

$$h(n:N_p) = 1 \qquad 1 \leq n \leq N_p/2$$

$$h(n:N_p) = 2 - 2n/N_p \qquad N_p/2+1 \leq n \leq N_p$$

and from Eq. (2.9), the frequency compression operation is given by

$$y(n) = s(n+N_p) + h(n:N_p)[s(n)-s(n+N_p)]$$

$$\text{for } n = 1,2,\ldots,N_p$$

For trapezoidal window function, this reduces to

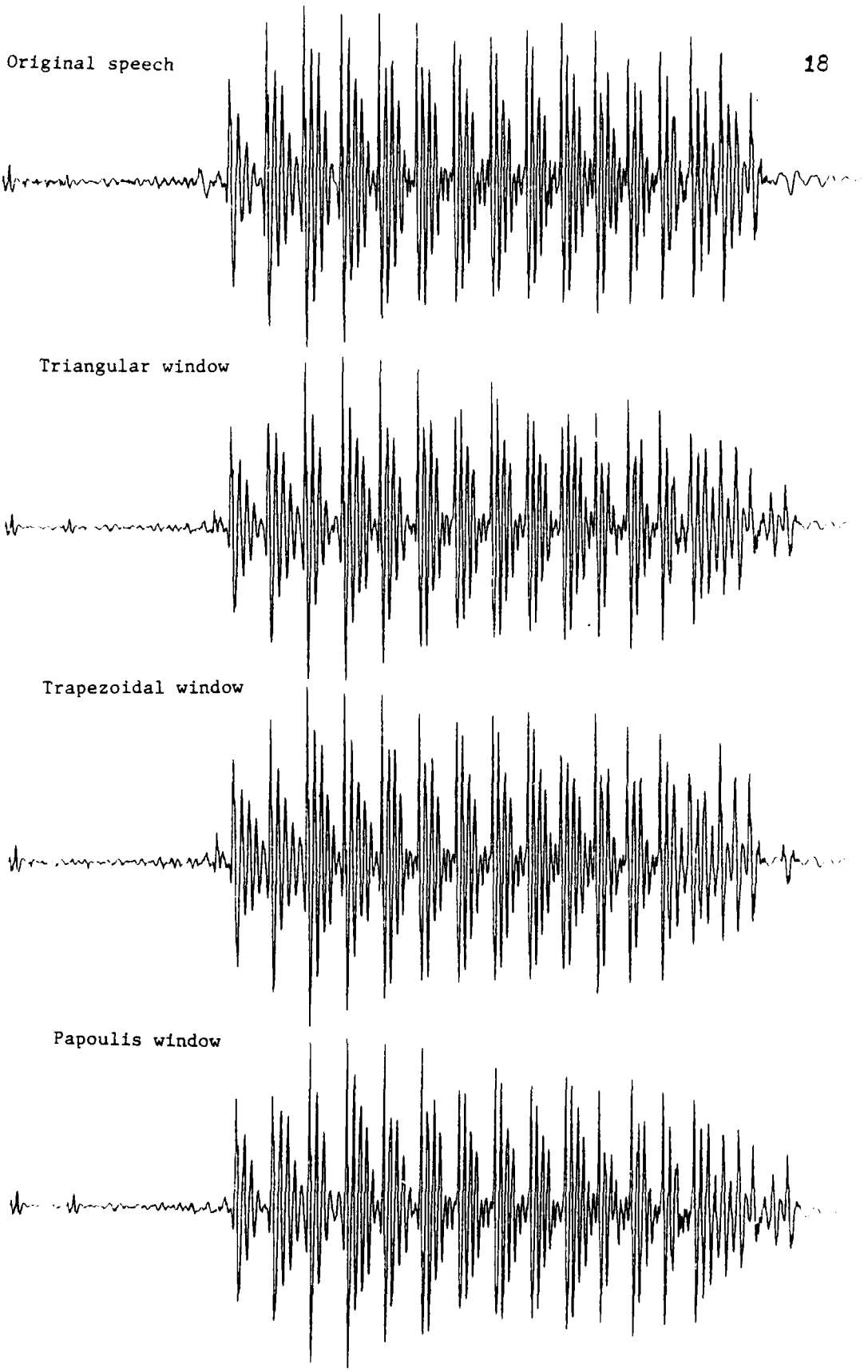
$$y(n) = s(n) \qquad 1 \leq n \leq N_p/2$$

$$y(n) = s(n+N_p) + h(n:N_p)[s(n)-s(n+N_p)]$$

$$N_p/2+1 \leq n \leq N_p$$

TABLE 2.1

Type	h(n:N_p)	# of Multiplications per pitch period	# of Additions per pitch period	SEGSNR in frequency domain
Triangular	$1 - \frac{n-1}{N_p-1}$	N_p	$1 + N_p$	11.39 dB
Cosine	$\text{Cos}\left[\frac{\pi(n-1)}{2(N_p-1)}\right]$	$2N_p$	$N_p + 1$	11.69 dB
Hanning	$\frac{1}{2} + \frac{1}{2}\text{Cos}\left[\frac{\pi(n-1)}{N_p-1}\right]$	$3N_p$	$2N_p + 1$	11.82 dB
Tukey	$\begin{cases} 1 & 1 \le n \le N_p/2 \\ \frac{1}{2}\left[1 - \text{Cos}\left(\frac{2\pi n}{N_p}\right)\right] & \frac{N_p}{2} + 1 \le n \le N_p \end{cases}$	N_p	$\frac{N_p}{2}$	11.24 dB
Popoulis	$\frac{1}{\pi}\text{Sin}\left[\frac{(n-1)\pi}{N_p - 1}\right] + \left[1 - \frac{n-1}{N_p-1}\right]\text{Cos}\left[\frac{\pi(n-1)}{N_p-1}\right]$	$3N_p$	$3N_p$	11.55 dB
Trapezoidal	$\begin{cases} 1 & 1 \le n \le \frac{N_p}{2} \\ 2 - \frac{2n}{N_p} & \frac{N_p}{2} + 1 \le n \le N_p \end{cases}$	$\frac{N_p}{2}$	$\frac{N_p}{2} + 1$	11.28 dB
Hamming	$0.54 + 0.46\ \text{Cos}\left[\frac{\pi(n-1)}{N_p-1}\right]$	$3N_p$	$2N_p + 1$	11.75 dB


Original speech
Triangular window
Trapezoidal window
Papoulis window

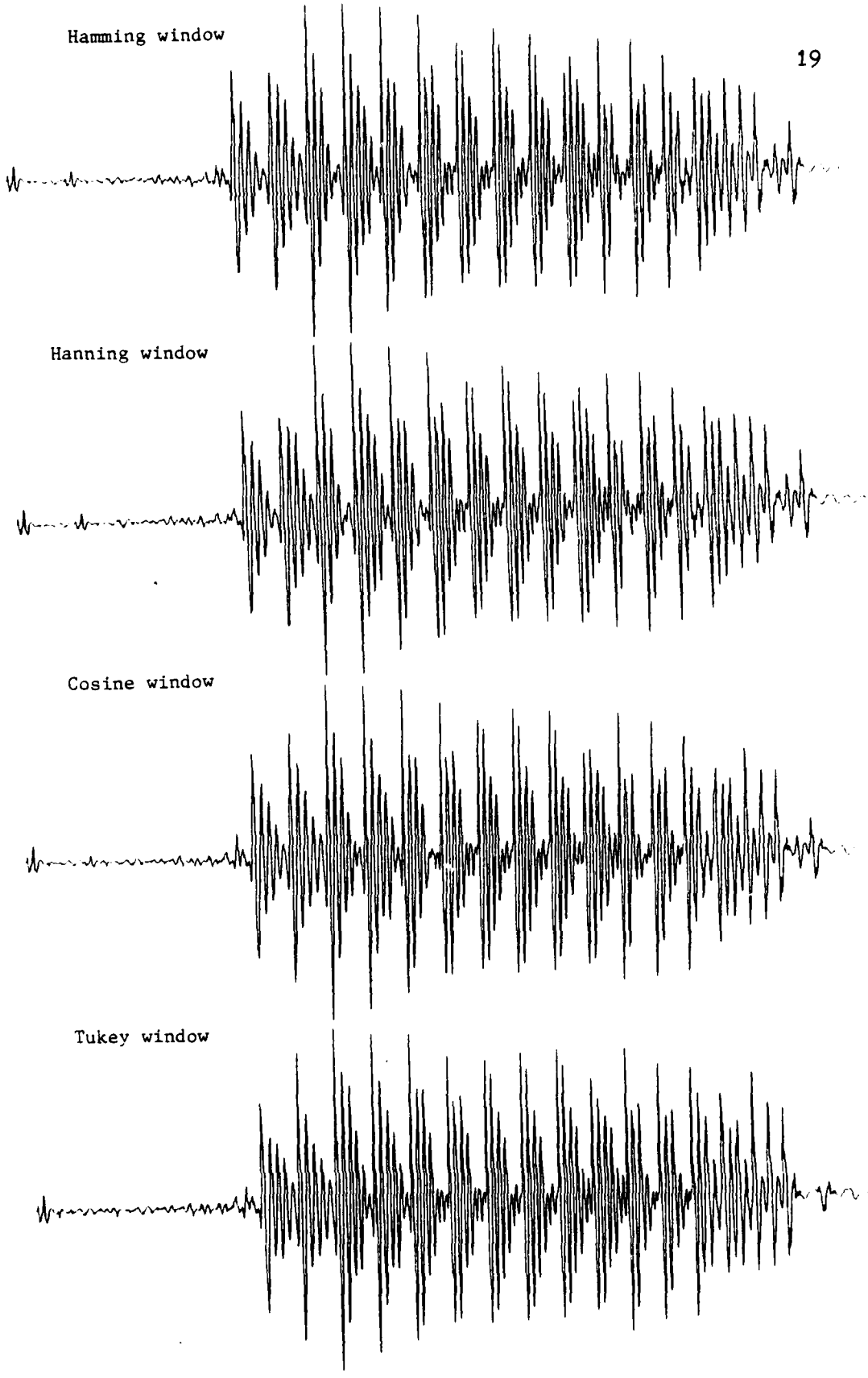


Fig. 2.6 TDHS output for the word "CATS" for different window functions.

From the comparisons of the above equations, it can be seen that it requires $N_p/2$ multiplications and N_p additions to produce N_p compressed speech samples for trapezoidal window as compared to N_p multiplications and $2N_p$ additions for triangular window.

The quality of output speech generated by using either the triangular or the trapezoidal window is almost the same. Therefore, the choice depends mainly upon the simplicity of implementation in hardware.

2.5 COMPRESSION RATIO

In previous sections, a compression factor of 2 was considered. However, other compression ratios are possible. As the value of this ratio increases, more interharmonic aliasing of pitch harmonics results. Such distortion could be tolerated in certain applications. In speech communication, speech quality is important and therefore, spectral distortions need to be kept small. A compression ratio of 2:1 is acceptable in this study. However, the distortion caused by compression and expansion process can be reduced by employing 3:2 compression.

Fig. 2.7 shows how three pitch periods can be compressed into two and expanded back into three. These operations can be put into equation form as follows. The frequency compressed speech, y(n) is given by

$$y(n) = s(n)\, h(n:2N_p) + s(n+N_p)[1-h(n:2N_p)] \quad \text{for } n = 1,2,\ldots,2N_p \qquad (2.12)$$

or

$$y(n) = s(n+N_p) + h(n:2N_p)[s(n)-s(n+N_p)] \quad n = 1,2,\ldots,2N_p \qquad (2.13)$$

where s(n) is the original speech samples, N_p is the pitch period expressed in terms of number of samples and $h(n:2N_p)$ is the window function given by

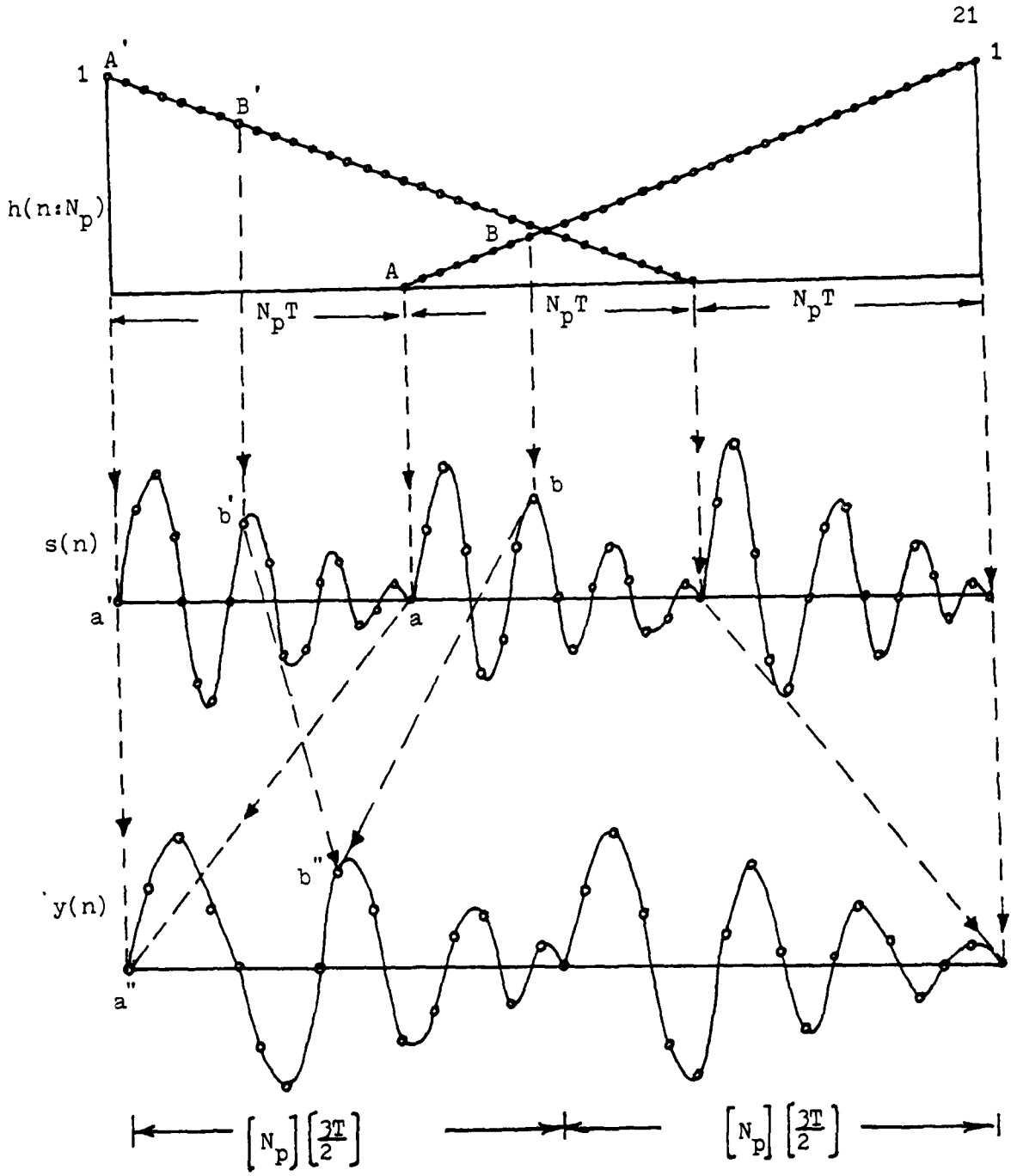


Fig. 2.7(a) TDHS frequency compression for the compression ratio of 3:2.

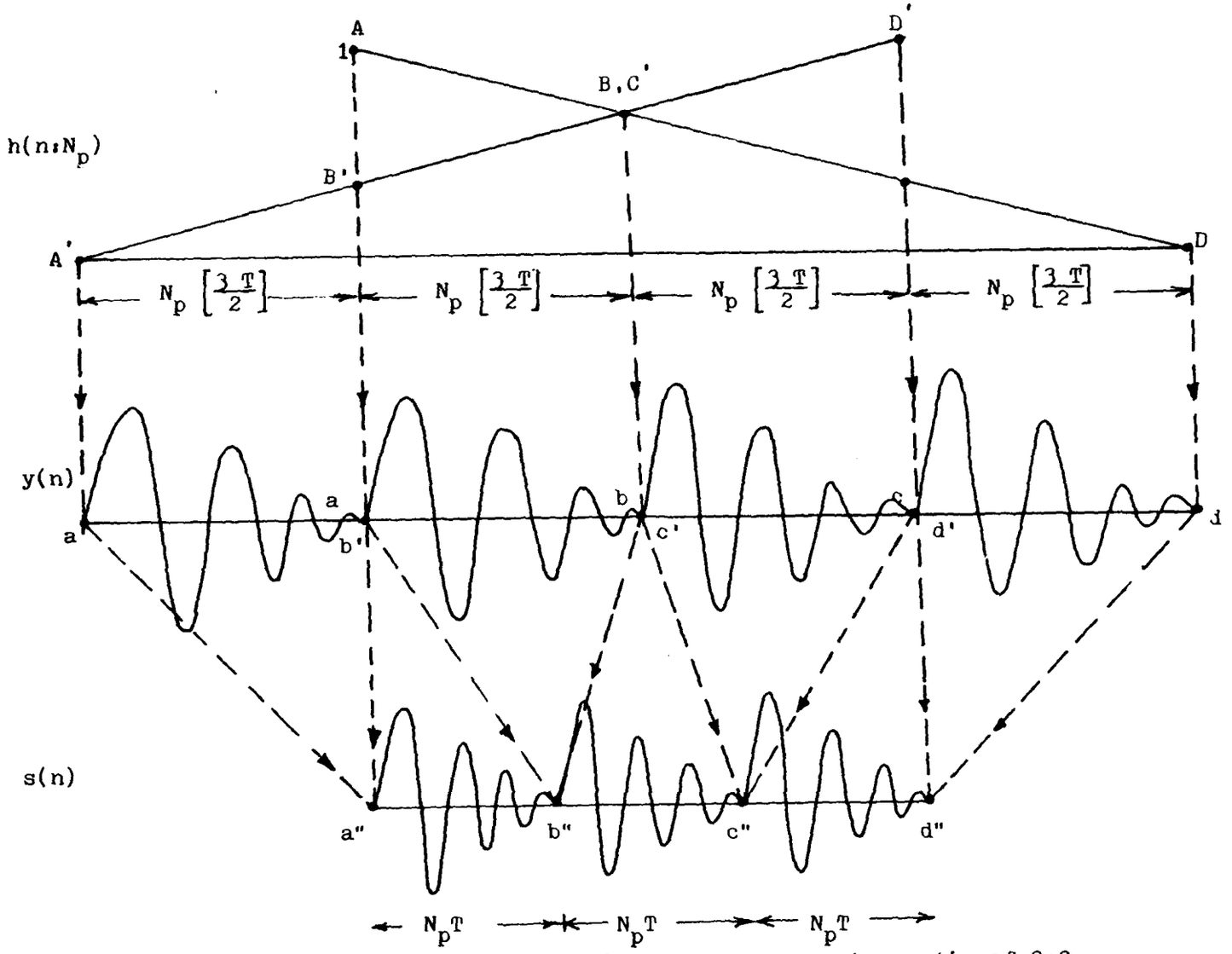


Fig. 2.7 (b) TDHS frequency expansion for the expansion ratio of 2:3.

$$h(n:2N_p) = 1 - \frac{(n-1)}{2N_p-1} \qquad 1 \leq n \leq 2N_p \qquad (2.14)$$

The frequency expansion operation, with the ratio of 2:3, on speech is performed similar to 1:2 as discussed earlier except for different window function and is given by

$$\hat{s}(n) = y(n)[1-h(n:3N_p)] + y(n+N_p)\, h(n:3N_p)$$

or

$$\hat{s}(n) = y(n) + h(n:3N_p)[y(n+N_p)-y(n)]$$

$$n = 1,2,\ldots,3N_p \qquad (2.15)$$

where

$$h(n:3N_p) = 1 - \frac{(n-1)}{3N_p-1} \qquad 1 \leq n \leq 3N_p \qquad (2.16)$$

Frequency scaling operations discussed above were simulated. The distortion found in the output speech was less than with the 2:1 compression scheme, as anticipated. This is evident from the results listed in Table 2.2.

TABLE 2.2

Comparison of 2:1 and 3:2 compression ratio scheme.

Sentence 1, Male speaker, Block size = 80 samples searching range: $20<T<100$

Type of Window	SEGSNR (Time domain)		SEGSNR (Frequency domain)	
	Compression Ratio 2:1	Compression Ratio 3:2	Compression Ratio 2:1	Compression Ratio 3:2
Triangular	8.06 dB	9.34 dB	11.39 dB	13.41 dB
Hanning	8.42 dB	10.31 dB	11.88 dB	15.01 dB
Tukey	7.30 dB	9.50 dB	11.24 dB	16.50 dB
Trapezoidal	7.48 dB	9.43 dB	11.28 dB	15.64 dB

The increase, in segmental SNR in frequency domain, as high as 5 dB could be obtained by employing 3:2 compression scheme. The increase in SEGSNR ususally indicates the improvement in speech quality. However, correlation between the extent of such improvement and the increase in SEGSNR is not known and is a separate topic of research [Barnwell, 1979].

Although this scheme looks promising, it has certain drawbacks. The number of bits per sample available for quantization is reduced significantly. For example, for the bit rate of 9.6 kbs and sampling rate of 6.4 kHz, the bits available per sample are reduced from 3 to 2.25 for 3:2 compression ratio scheme. Such reduction in available entropy, not only leads to more quantization noise, but makes fewer bits available for error protection, thus making the system more susceptible to channel noise.

For robust 9.6 kbs system, a compression ratio of 2:1 was thought to be a good choice. However, for higher bit rates and/or less noisy channel, a compression scheme of 3:2 would be an attractive choice. Should the pitch periods (N_p) form the side information, a 2:1 scheme would require transmitting the pitch information every $2N_p$ samples as against $3N_p$ samples in a 3:2 scheme. Therefore, a 3:2 scheme would require 33% less bit rate to transmit the pitch than for a 2:1 scheme.

2.6 PITCH EXTRACTION

In earlier sections, it was shown that the TDHS algorithm consists of properly weighing several adjacent input signal segments with pitch dependent duration by a suitable window function, to produce an output segment. In the frequency domain, the time-domain operations are equivalent to shifting the individual pitch harmonics of the quasi-periodic voiced-speech signal according to the center frequency of the subband in which

each harmonic component is located. The number of subbands into which the speech band is divided is pitch dependent. The pitch adaptive nature of the algorithm requires a pitch extraction operation in the system. The choice of a method to be used for extraction depends on how accurate the pitch estimation needs to be.

Since the TDHS algorithm is pitch adaptive, an error in the pitch estimation may cause a distortion in the output speech. Malah, in his recent work [1981], has given the upperbound for an error in the pitch period estimation for the exact reconstruction of the signal harmonics after the frequency compression followed by the frequency expansion. The upperbound, given by Malah, is inversely proportional to the compression ratio and the maximum frequency to be reconstructed exactly. For fairly good quality speech reproduction, at least the second formant frequency of the voiced speech should be reconstructed correctly. This fact can crudely be verified by listening to the speech which is passed through a low-pass filter with different cutoff frequencies. The second formant frequencies for most of the vowels are located below 1.5 to 2 KHz [Rabiner and Schafer, 1978]. With the compression ratio of 2, the allowed pitch-period error for the above frequencies become 0.16 and 0.125 msec respectively. With sampling rate of 6400 Hz, this is equivalent to 0.8 to 1 sample error. In the simulation studies, it was noticed that a pitch error of twice this amount could be tolerated which confirms with Malah's observation [1981].

Several pitch extraction techniques exist in the literature [Gold & Rabiner 1969; Ross, et. al 1974; Sondhi 1968; Rabiner 1977]. Many of these techniques have some form of a logic for making a voiced/unvoiced

(V/UV) decision as well as for the pitch data smoothing. The algorithm presented here does not need such a complex technique and therefore, only simple techniques will be studied. Two such techniques are autocorrelation [Rabiner, 1977] and AMDF [Ross, et. al 1974]. The pitch was estimated by using the above methods for original speech, center clipped speech [Sondhi, 1968], 3-value center clipped speech [Dubnowski, 1976] and 2-value clipped speech. The autocorrelation and AMDF methods, combined with the four different types of speech input, form essentially eight different techniques. The methods and the results are discussed in the following paragraphs.

Simple methods such as autocorrelation and AMDF, were tried for pitch estimation. These methods estimate the pitch periods quite accurately in the voiced segments of speech, except for double or triple pitch picking. The double pitch-period corresponds to performing the filter bank analysis with twice as many filters. This means that only every other filter contains a pitch harmonic. This algorithm does not need any voiced-unvoiced decision. All these reasons made it possible to use a simple pitch extraction method.

The autocorrelation method involves forming the short time autocorrelation function as in Eq. (2.17)

$$R(\ell) = \sum_{m=o}^{N-1} s(m)\, s(m+\ell) \qquad T_{min} \leq \ell \leq T_{max} \qquad (2.17)$$

The autocorrelation function representation of the signal is a convenient way of displaying certain properties of the signal. For example, if the signal is periodic with period equal to T samples, it can be shown that

$$R(\ell) = R(\ell+T)$$

and $R(\ell)$ attains its maximum value at $\ell = 0, \pm T, \pm 2T, \ldots$ The pitch determination involves computing $R(\ell)$ as in Eq. (2.17) for different lags ℓ and locating the maximum. To check the periodicity of the waveform one needs to check at least two periods. Therefore, the searching range for ℓ is of the order of two pitch periods. The blocksize N should be chosen to give a good indication of the changing properties of the speech signal. A block size on the order of a pitch period was found to be a good choice.

The above method, though simple to implement, involves extensive computations. For example, if the block length is N and the searching range is r, then for each value of r there are N multiplications and N additions. Hence the total number of multiplications for finding the pitch period becomes $N \cdot r$; if r=150 and N=80 then this number is 12000. This is just for one block of samples. This many multiplications consumes significant processing time which may cause problem in a real-time implementation. This led to a search for another technique which is computationally simpler and yet provides accurate results.

This is possible by the use of Average Magnitude Difference Function (AMDF). This technique is based upon the idea that for a truly periodic input of period T, the sequence

$$d(n) = s(n+k) - s(n)$$

would be zero for $k = 0, \pm T, \pm 2T \ldots$ For a short-segment of voiced speech, it is reasonable to expect that d(n) will be small at multiples of the period, but not identically zero. The short-time average magnitude of d(n) as a function of k should be small whenever k is close to the period. The short-time AMDF [Ross, et. al, 1974] is thus defined as

$$A(k) = \sum_{n=0}^{N-1} |s(n+k) - s(n)| \qquad (2.18)$$

$$T_{min} \leq k \leq T_{max}$$

The considerations for the choice of blocksize and the searching range is the same as discussed above. The AMDF function is implemented with subtraction, addition and absolute value operations, in contrast to addition and multiplication operations for the autocorrelation function. With floating point arithmetic, where multiplies and adds take approximately the same time, about the same time is required for either method with the same window length. However, for special purpose hardware, or with fixed point arithmetic, the AMDF appears to have an advantage. In this case, multiplies usually are more time consuming and furthermore either scaling or a double precision accumulator is required to hold the sum of lagged products.

One of the major limitations of the autocorrelation representation is that in a sense it retains too much of the information in the speech signal. As a result, the autocorrelation function has many peaks. Most of these peaks can be attributed to the damped oscillations of the vocal tract response which are responsible for the shape of each period of speech wave. Usually the peak at the pitch period has the greatest amplitude (smallest in case of AMDF). However, rapidly changing format frequencies can cause bigger autocorrelation peaks than those due to the periodicity of the vocal excitation. In such cases, the simple procedure of picking the largest peak in autocorrelation will fail.

To avoid this problem it is again useful to process the speech signal so as to make the periodicity more prominent while suppressing other

distracting features of the signal. Techniques which perform this type of operation on a signal are sometimes called "spectrum flatteners" since their objective is to remove the effects of the vocal tract transfer function, thereby bringing each harmonic to the same amplitude level as in the case of a periodic impulse train. Numerous spectrum flattening techniques have been proposed [Sondhi, 1968]; one technique is called "center clipping" [Sondhi, 1968], and is obtained by a nonlinear transformation

$$y(n) = C[x(n)]$$

where C[ ] is shown in Fig. 2.8(a). The operation of center clipper is depicted in Fig. 2.8(b). From a block of speech samples the absolute maximum amplitude S_{max} is found; the clipping level C_L is set equal to a fixed percentage of S_{max}. It can be seen that for samples above C_L, the output of the center clipper is equal to the input minus the clipping level. For samples below the clipping level, the output is zero.

Clearly, setting the clipping level is important. If the clipping level is large, only a small number of peaks will exceed the clipping level and only a few undesirable pulses will occur in the output. The clearest indication of periodicity is obtained for the highest possible clipping level. There is, however, a difficulty with using a too high a clipping level. It is possible that the amplitude of the signal may vary appreciably across the duration of the speech segment (e.g. at the beginning or end of voicing) so that if the clipping level is set at a high percentage of the maximum amplitude across the whole segment, there is a possibility that much of the waveform will fall below the clipping level and be lost. In the simulation studies, it was found that such situation

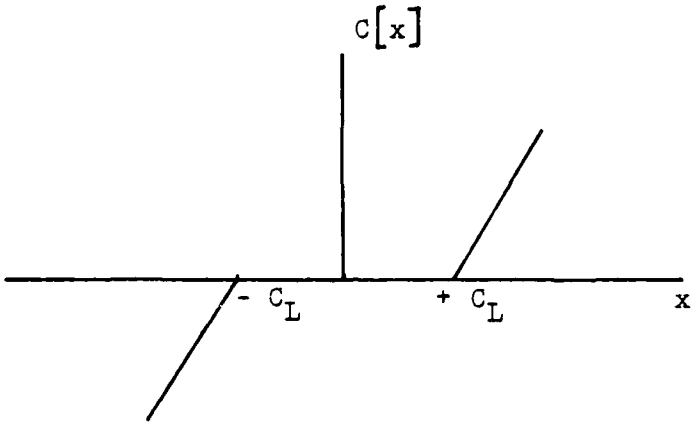


Fig. 2.8 (a) Center clipping function

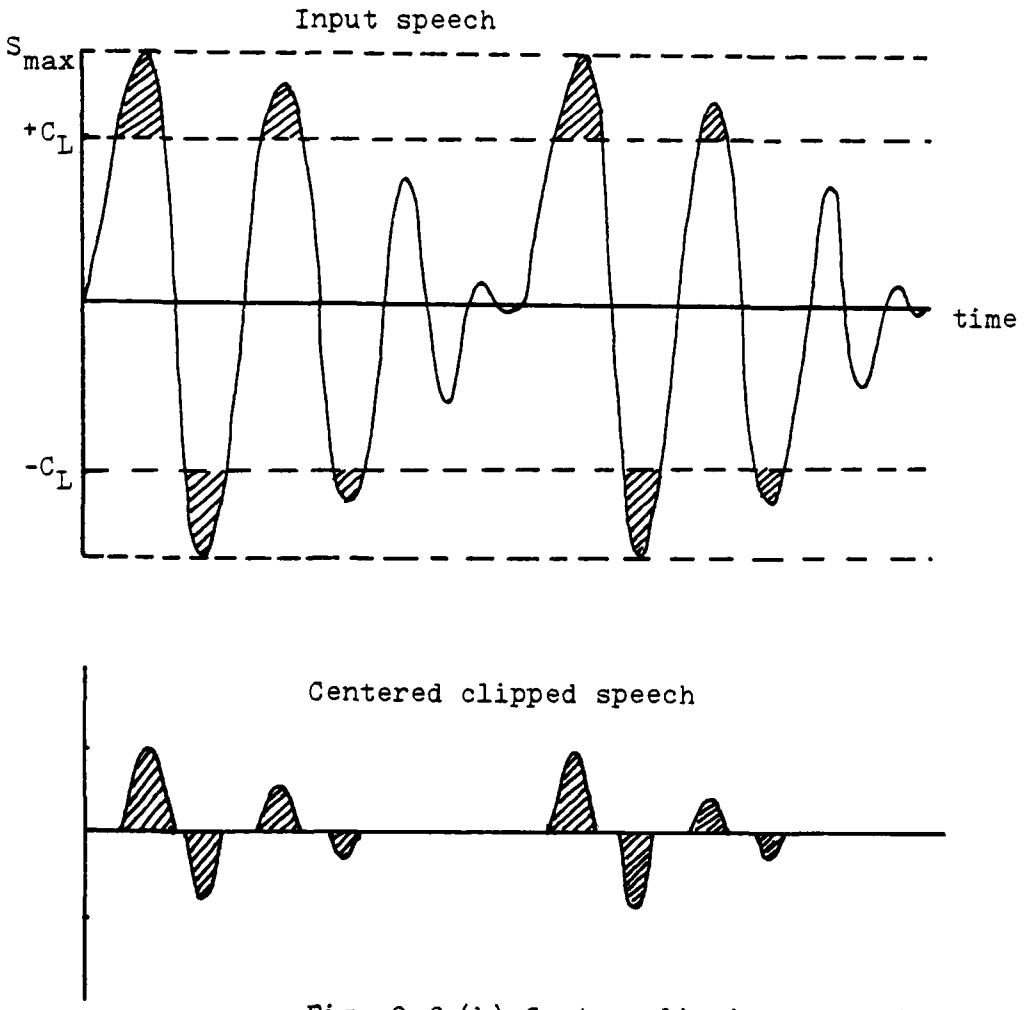


Fig. 2.8 (b) Center clipping operation.

is avoided if clipping level is kept around 30%. This same observation was also made by Sondhi [1968]. If the clipping level is more than 60% the situation noted above usually does occur and the estimation of the pitch period is in error. This is shown in Table 2.3 by the arrows pointed to the wrong pitch periods. A procedure [Rabiner and Schafer, 1978] which permits a greater percentage (60-80%) to be used is to find the peak amplitude in both the first third and the last third of the segment and set the clipping level at a fixed percentage of the minimum of these two maximum levels. This procedure was incorporated in the pitch extraction algorithm and results are shown in Table 2.3. The table shows the analysis intervals and the corresponding pitch periods. The analysis interval is chosen to be 200 samples. The first and the last sample number of this interval is obtained as follows. Suppose the first sample number of an analysis interval is n. The last sample number of the interval becomes n+199 to make the length equal to 200. Let the pitch period for the analysis interval (n to n+199) be N_p. The next interval becomes ($n+N_p$ to $n+2N_p$ +199). Since the pitch is different for different pitch extraction techniques in unvoiced segment of speech, the analysis interval differs. This can also happen due to double or triple pitch picking in the voiced speech. Comparing the results in Table 2.3 and it can be seen that pitch errors due to high clipping level are corrected by using the procedure outlined above. Note that the double or triple pitch periods are not considered pitch errors because they do not degrade the performance.

A simple modification of the center clipping function leads to a great simplification in computation of the autocorrelation function with

TABLE 2.3

The effect of clipping level on the pitch extraction for female speaker.

C_L = 30 %		C_L = 60 %		C_L = 60 % With improved procedure.	
Sample #	Pitch	Sample #	Pitch	Sample #	Pitch
7779 - 7978	2Ø	7769 - 7968	2Ø	7781 - 798Ø	72
7819 - 8Ø18	25	7809 - 8ØØ8	2Ø	7925 - 8124	77
7869 - 8Ø68	25	7849 - 8Ø48	2Ø	8Ø79 - 8278	2Ø
7919 - 8118	51	7889 - 8Ø88	77	8135 - 8334	29
8Ø21 - 822Ø	26	8Ø43 - 8242	29	8193 - 8392	29
8Ø73 - 8272	2Ø	81Ø1 - 83ØØ	2Ø	8251 - 845Ø	29
8129 - 8328	29	8141 - 834Ø	2Ø	83Ø9 - 85Ø8	29
8187 - 8386	29	8181 - 838Ø	29	8367 - 8566	3Ø
8245 - 8444	29	8239 - 8438	29	8427 - 8626	9Ø
83Ø3 - 85Ø2	29	8297 - 8496	29	86Ø7 - 88Ø6	3Ø
8361 - 856Ø	3Ø	8355 - 8554	3Ø	8667 - 8866	3Ø
8421 - 862Ø	3Ø	8415 - 8614	3Ø	8727 - 8926	3Ø
8481 - 868Ø	3Ø	8475 - 8674	3Ø	8787 - 8986	3Ø
8541 - 874Ø	3Ø	8535 - 8734	3Ø	8847 - 9Ø46	3Ø
86Ø1 - 88ØØ	3Ø	8595 - 8794	3Ø	89Ø7 - 91Ø6	91
8661 - 886Ø	3Ø	8655 - 8854	3Ø	9Ø89 - 9288	92
8721 - 892Ø	3Ø	8715 - 8914	3Ø	9273 - 9472	92
8781 - 898Ø	3Ø	8775 - 8974	3Ø	9457 - 9656	31
8841 - 9Ø4Ø	3Ø	8835 - 9Ø34	3Ø	9519 - 9718	31
89Ø1 - 91ØØ	91	8895 - 9Ø94	91	9581 - 978Ø	31
9Ø83 - 9282	31	9Ø77 - 9276	92	9643 - 9842	31
9145 - 9344	31	9261 - 946Ø	61	97Ø5 - 99Ø4	61
92Ø7 - 94Ø6	31	9383 - 9582	3Ø	9827 - 1ØØ26	31
9269 - 9468	92	9443 - 9642	31	9889 - 1ØØ88	91
9453 - 9652	31	95Ø5 - 97Ø4	31	1ØØ71 - 1Ø27Ø	3Ø
9515 - 9714	31	9567 - 9766	31	1Ø131 - 1Ø33Ø	91
9577 - 9776	31	9629 - 9828	31	1Ø313 - 1Ø512	61
9639 - 9838	31	9691 - 989Ø	3Ø	1Ø435 - 1Ø634	3Ø
97Ø1 - 99ØØ	31	9751 - 995Ø	31	1Ø495 - 1Ø694	3Ø
9763 - 9962	61	9813 - 1ØØ12	3Ø	1Ø555 - 1Ø754	91
9885 - 1ØØ84	61	9873 - 1ØØ72	91		
1ØØØ7 - 1Ø2Ø6	91	1ØØ55 - 1Ø254	91		
1Ø189 - 1Ø388	61	1Ø237 - 1Ø436	→ 2Ø		
1Ø311 - 1Ø51Ø	31	1Ø277 - 1Ø476	→ 2Ø		
1Ø373 - 1Ø572	91	1Ø317 - 1Ø516	3Ø		
1Ø555 - 1Ø754	91	1Ø377 - 1Ø576	3Ø		
		1Ø437 - 1Ø636	3Ø		
		1Ø497 - 1Ø696	3Ø		

essentially no degradation in utility for pitch detection [Rabiner, Schafer, Dubnowski, 1976]. This modification is shown in Fig. 2.9. As indicated there, the output of the clipper is +1 if $x(n) > C_L$ and is -1 if $x(n) < -C_L$. Otherwise the output is zero. Although this operation tends to emphasize the importance of peaks that just exceed the clipping level, the autocorrelation function is very similar to that of the center clipper of Fig. 2.8.

The computation of the autocorrelation function for a 3-level center clipped signal is particularly simple. The product $s(k)s(k+\ell)$ in Eq. (2.17) can have only three values.

$$s(k)sk+\ell) = \begin{cases} 0 & \text{if } s(k) = 0 \text{ or } s(k+\ell) = 0 \\ +1 & \text{if } s(k) = s(k+\ell) \\ -1 & \text{if } s(k) \neq s(k+\ell) \end{cases}$$

Thus, in hardware terms, only simple combinatorial logic and an up-down counter is required to accumulate the autocorrelation value for each value of ℓ. Likewise, the input to the AMDF algorithm could also be center clipped speech.

The two-level or infinite clipper shown in Fig. 2.10, was tried for a hardware simplification to the 3-level clipper described above. However, the pitch period estimation was not accurate. The explanation could be derived from Licklider and Pollack's [1948] experiment. They showed that, whereas speech that has been infinitely peak clipped is highly intelligible, even a few percent of center clipping drastically reduces intelligibility. The reason is infinite peak clipping retains the formants of the speech signal (although it introduces a few secondary "formants"), center

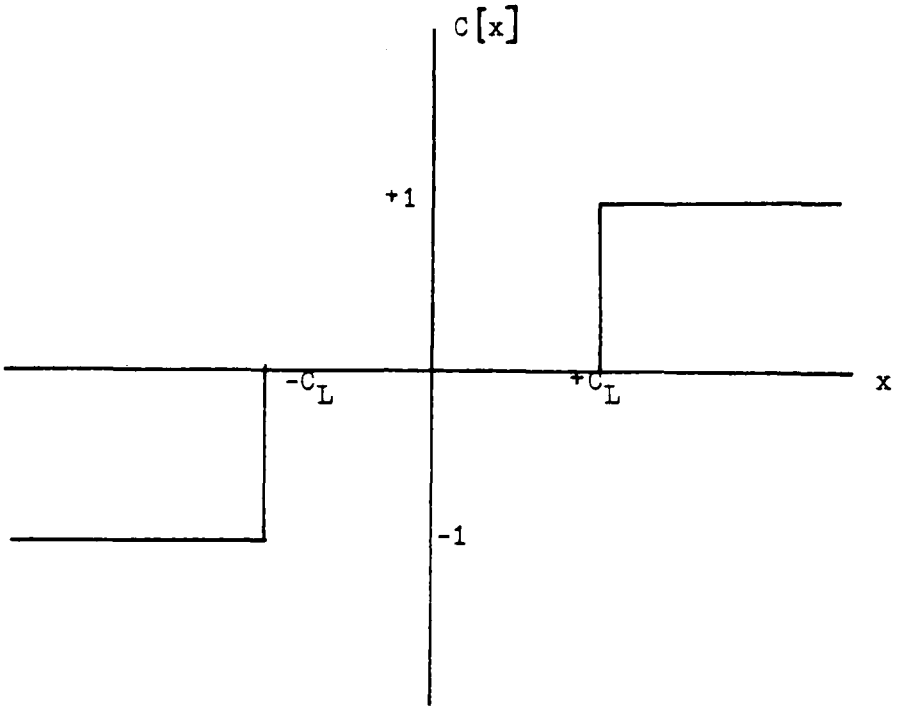


Fig. 2.9 3-level center clipping function

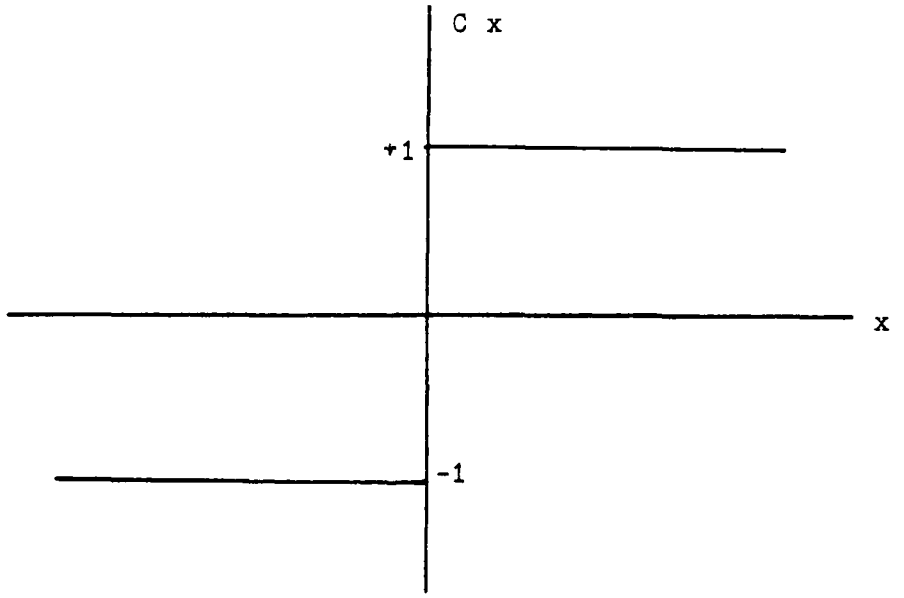


Fig. 2.10 2-level center clipping function

clipping destroys the formant structure, while retaining the periodicity. It is the removal of formant structure that is so important for a good pitch extractor.

Table 2.4 shows the output of the pitch extractor for seven different techniques. The one which uses three level center clipper was found to be accurate enough for the desired applications and hence was selected for its hardware simplicity. Note that, as discussed earlier, the analysis intervals differ for the different pitch estimators. Hence, the care must be taken to compare the pitch period variations for the entire segment of voiced speech instead of one particular analysis interval.

The hardware simplicity of the pitch detector becomes quite important because of the unique feature of this algorithm. The pitch is extracted at the receiver from the reconstructed compressed speech. The two pitch extractors, one at the transmitter and another at the receiver, increase the cost of the system hardware. However, for a 3-level center clipped autocorrelation method, hardware requirements are minimum and hence, the hardware cost is marginal. The pitch extraction at the receiver becomes a little more difficult because the input speech for the pitch detector is noisy compressed speech, possibly with channel errors. The noise in the reconstructed compressed speech is the quantization noise which is small enough to not cause a pitch detection error. However, a smaller number of pitch periods per unit time are available in the compressed speech. This is the major problem to extracting pitch at the receiver. The problem could become serious for male speakers because of the small value of the fundamental frequency. The pitch extraction techniques outlined above can still be used at the receiver. However, a judicious choice of the

TABLE 2.4

Pitch extracted using different pitch estimation techniques. Sentence 11, Female speaker
Block size= 50, Searching range = (20,100), Sampling frequency = 6400 Hz.

AUTO CORRELATION		AMDF		AUTO CORRELATION		AMDF		AUTO CORRELATION		AMDF		AUTO CORRELATION	
Original speech		original speech		center clipped speech		center clipped speech		3-value center clipped		3-value center clipped		2-value center clipped speech	
sample #	Np	sample #	Np	sample #	Np	sample #	Np	sample #	Np	sample #	Np	sample #	Np
5127 - 5326	51	5067 - 5256	66	5153 - 5352	20	5039 - 5238	20	5165 - 5364	20	5003 - 5202	32	5097 - 5296	20
5229 - 5428	79	5187 - 5386	23	5193 - 5392	00	5095 - 5294	91	5205 - 5404	29	5067 - 5266	76	5163 - 5362	20
5307 - 5506	30	5233 - 5432	23	5363 - 5562	29	5277 - 5476	27	5263 - 5462	26	5219 - 5418	23	5193 - 5392	66
5447 - 5646	30	5279 - 5478	27	5411 - 5610	30	5331 - 5530	29	5315 - 5514	20	5265 - 5464	26	5303 - 5502	27
5507 - 5706	30	5333 - 5532	29	5471 - 5670	30	5389 - 5588	29	5371 - 5570	29	5317 - 5516	20	5367 - 5556	29
5567 - 5766	30	5391 - 5590	30	5531 - 5730	30	5447 - 5646	30	5429 - 5620	30	5373 - 5572	29	5415 - 5614	30
5627 - 5826	30	5451 - 5650	30	5591 - 5790	60	5507 - 5706	30	5489 - 5688	30	5431 - 5630	30	5476 - 5674	30
5687 - 5886	90	5511 - 5710	30	5711 - 5910	90	5567 - 5766	30	5549 - 5748	30	5491 - 5690	30	5535 - 5734	30
5867 - 6066	90	5571 - 5770	30	5891 - 6090	90	5627 - 5826	30	5609 - 5808	30	5551 - 5750	30	5596 - 5794	60
6047 - 6246	09	5631 - 5830	30	6071 - 6270	90	5687 - 5886	30	5669 - 5868	30	5611 - 5810	30	5716 - 5914	30
6226 - 6424	30	5691 - 5890	30	6251 - 6450	30	5747 - 5946	30	5729 - 5928	30	5671 - 5870	30	5775 - 5974	60
6286 - 6484	30	5751 - 5950	30	6311 - 6510	30	5807 - 6006	30	5789 - 5988	30	5731 - 5930	30	5895 - 6094	30
6346 - 6544	30	5811 - 6010	30	6371 - 6570	30	5867 - 6066	30	5849 - 6048	30	5791 - 5990	30	5956 - 6154	30
6405 - 6604	30	5871 - 6070	30	6431 - 6630	30	5927 - 6126	30	5909 - 6108	30	5851 - 6050	30	6015 - 6214	30
6465 - 6664	30	5931 - 6130	30	6491 - 6690	30	5987 - 6186	30	5969 - 6160	30	5911 - 6110	30	6075 - 6274	30
6525 - 6724	30	5991 - 6190	30	6551 - 6750	30	6047 - 6246	30	6029 - 6228	30	5971 - 6170	30	6135 - 6334	60
6585 - 6784	30	6051 - 6250	30	6611 - 6810	30	6107 - 6306	30	6089 - 6288	30	6031 - 6230	30	6253 - 6452	30
6645 - 6844	30	6111 - 6310	30	6671 - 6870	30	6167 - 6366	30	6149 - 6348	30	6091 - 6290	30	6313 - 6512	30
6705 - 6904	30	6171 - 6370	30	6731 - 6930	30	6227 - 6426	30	6209 - 6408	30	6151 - 6350	30	6373 - 6572	30
6765 - 6964	31	6231 - 6430	30	6791 - 6990	31	6287 - 6486	30	6269 - 6468	30	6211 - 6410	30	6433 - 6632	30
6827 - 7026	32	6291 - 6490	30	6853 - 7052	31	6347 - 6546	30	6329 - 6528	30	6271 - 6470	30	6493 - 6692	09
6891 - 7090	22	6351 - 6550	30	6915 - 7114	72	6407 - 6606	30	6389 - 6588	30	6331 - 6530	30	6671 - 6870	30
6935 - 7134	51	6411 - 6610	30	7859 - 7250	70	6467 - 6666	30	6449 - 6648	30	6391 - 6590	30	6731 - 6930	30
7037 - 7236	66	6471 - 6670	30			6527 - 6726	30	6509 - 6708	30	6451 - 6650	30	6791 - 6990	31
		6531 - 6730	30			6587 - 6786	30	6569 - 6768	30	6511 - 6710	30	6853 - 7052	27
		6591 - 6790	30			6647 - 6846	30	6629 - 6828	30	6571 - 6770	30	6907 - 7106	44
		6651 - 6850	30			6707 - 6906	30	6689 - 6888	30	6631 - 6830	30	6995 - 7194	32
		6711 - 6910	30			6767 - 6966	31	6749 - 6948	31	6691 - 6890	30	7859 - 7250	29
		6771 - 6970	31			6829 - 7028	32	6811 - 7010	31	6761 - 6958	30		
		6833 - 7032	32			6893 - 7092	06	6873 - 7072	32	6811 - 7010	32		
		6897 - 7096	06			7863 - 7262	37	6937 - 7136	51	6875 - 7074	66		
		7867 - 7266	01					7039 - 7238	22	7007 - 7206	91		
								7083 - 7282	27				

blocklength and the searching range is necessary. At the transmitter, the blocklength of 80 and searching range of (20,100) samples was found to be a good choice. At the receiver, the blocklength of 50 and the same searching range was found to be satisfactory. These conclusions are based on the results obtained for three sentences. A study for several utterances might be necessary to come up with the best value for the blocklength and the searching range.

Pitch detection at the receiver avoids the transmission of pitch as a side information, thus, making the blocking of the quantizer data and the block synchronization unnecessary. However, channel noise can affect the compressed speech appreciably which inturn would cause wrong pitch estimation. For a bit-error-rate (BER) of 1%, the above situation does occur quite frequently but occurs less frequently if error protection is provided for the quantizer levels. In the simulation studies, the speech quality for the noisy channels with or without pitch transmission was found to be the same. This may be due to the masking effect of the distortion due to the errors in the quantizer levels on the distortion caused by the pitch errors. The results and further discussion of this are postponed until Chapter 4.

2.7 SUMMARY

A time domain harmonic scaling of speech was found to be an effective approach to represent speech with fewer number samples while maintaining excellent quality. A simple triangular window function was used for compression and expansion operations. A compression ratio of 2:1 was found to be a good choice. A 3-value center clipped autocorrelation technique

without any decision logic, V/UV decision or pitch data smoothing was found to be adequate for the purpose. The pitch is extracted both at the transmitter and the receiver. The harmonic distortion produced by TDHS operations could not be noticed due to masking properties of the ear. The TDHS output is coded using an ADPCM technique which is described in the next chapter.

CHAPTER 3

ADAPTIVE RESIDUAL CODER

3.1 INTRODUCTION

In the previous chapter, the compression and expansion operations performed at the transmitter and at the receiver respectively were discussed. Compressed speech, which consists of half the number of original samples for 2:1 compression ratio, needs to be coded and transmitted on the channel. This can be achieved by employing various schemes. Compressed speech is formed by using long-term redundancy in a manner different from the APC technique. Since some of the long-term redundancy in the original speech is already removed in forming the compressed speech, use of APC coder to code it would be inefficient. However, compressed speech could be effectively coded by exploiting the short term redundancy. This is done by using ADPCM coders.

The Adaptive Residual Coder (ARC) System, developed by Cohn and Melsa, [Sept. 1975] is an improved ADPCM system. The following sections describe the system structure of ARC and outline its design and performance for the compressed speech as an input signal. Various performance measures were tested for the ARC system and are also discussed in this chapter.

3.2 THE RESIDUAL ENCODER STRUCTURE

Fig. 3.1 shows the basic ADPCM structure augmented with dashed lines to indicate the flow of information for adaptation in the residual encoder. The underlying design principle of the adaptation procedure is

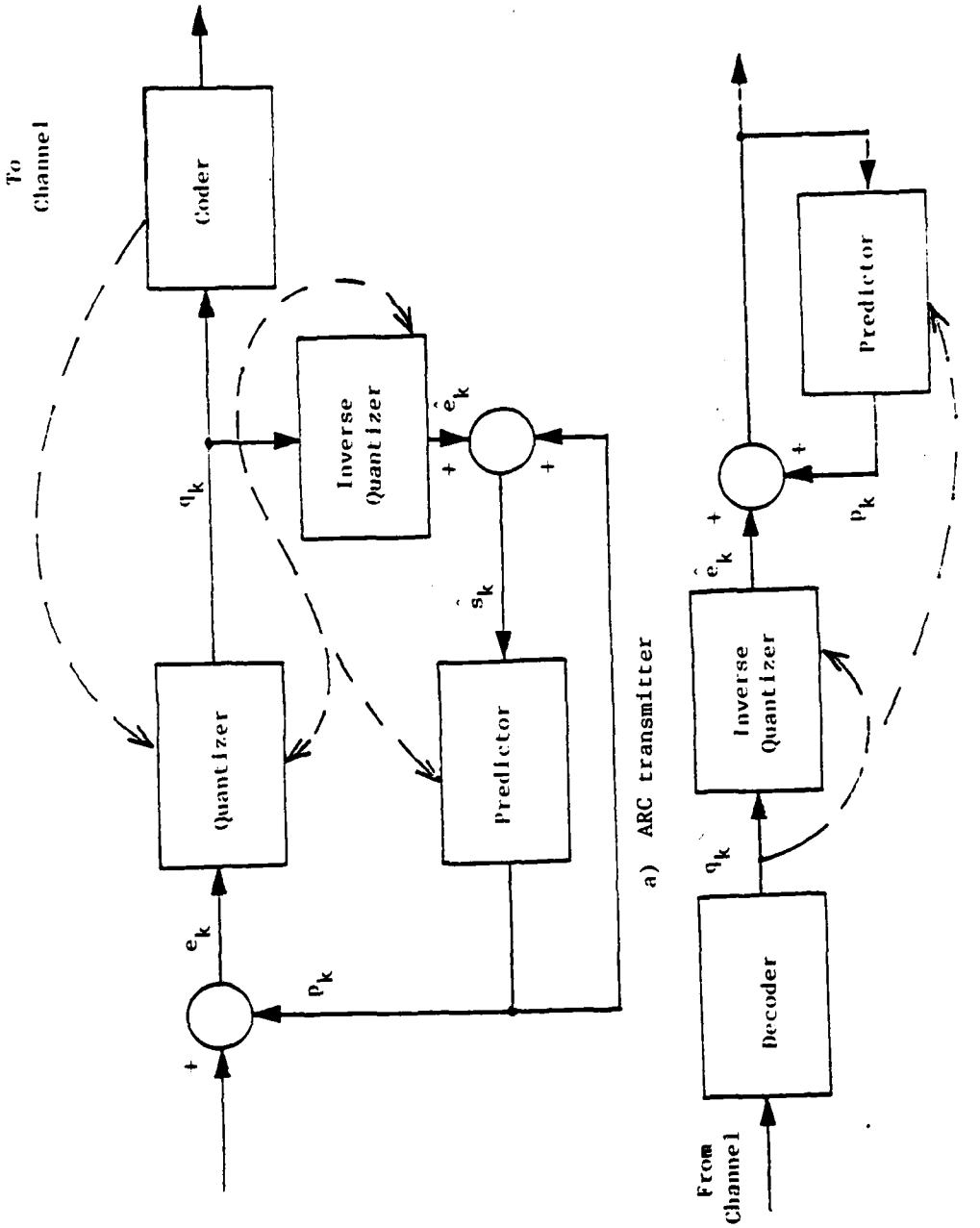


Fig. 3.1 Adaptive Residual Coder (ARC) System

that all information used in updating the quantizer and predictor be available both at the transmitter and the receiver. Since the only information sent from the transmitter to the receiver is the quantizer output q_k, all adaptation procedures must use quantities derivable from the q_k.

As mentioned earlier, the compressed speech samples y_k form the input to the ARC system. The sampling rate of y_k depends on the sampling frequency of original speech and the compression ratio employed in harmonic scaling operations. For example, if the input speech bandwidth is 3200 Hz and it is sampled at the Nyquist rate, and with a 2:1 compression ratio, then the sampling frequency of y_k becomes 3200 Hz and the bandwidth is compressed into 1600 Hz.

The system shown in Fig. 3.1 includes the pitch compensating adaptive quantizer. The object of an adaptive quantizer is to match the quantizer to the local statistics of the speech rather than to global or "average" statistics. Thus, if the speaker is talking loudly, a quantizer with a wide range should be used; but if the speaker is talking quietly, a narrow range should be used. In fact, the dynamic range of human speech varies significantly from syllable to syllable.

The usual method of quantizer adaptation is to adjust the range of the quantizer as a function of the prior quantizer output. Consider the quantizer structure shown in Fig. 3.2. Here the input is normalized by a state variable σ_k and then quantized with a fixed quantizer. If the expected range of the input signal is large, σ_k should be large; if the expected range is small, σ_k should be small. If the prior quantizer input was large, then from the correlation between successive samples, one might

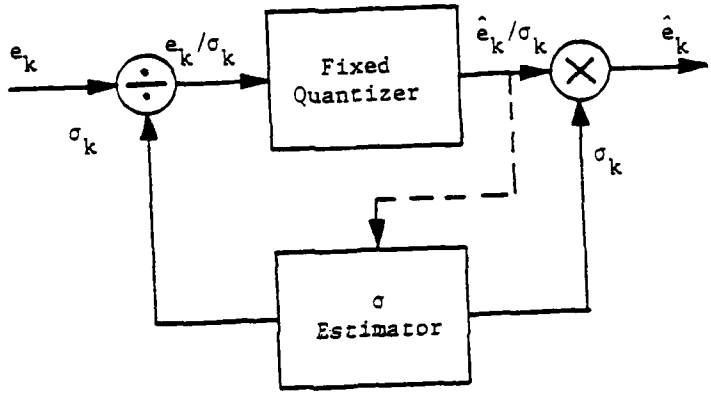


Fig. 3.2 Adaptive Quantizer

expect the next input to be large as well. Therefore, if the quantizer output is large, σ_k should be increased. Similarly, if the quantizer output is small, σ_k should be decreased.

Two different approaches have been suggested for quantizer adaptation: forward adaptation and backward adaptation. In a forward scheme, the adaptation decision is based on unquantized data and is communicated to the receiver as side information. Performance tests indicate (Noll, 1974) that the forward method is slightly better if no cost is assessed for the side information. Practical considerations, however, seem to favor backward adaptation.

Several investigators have evaluated adaptive PCM and DPCM systems that incorporate adaptive quantizer (Cummisky, et al., 1973; Gibson, et al., 1974; Castellino, et al., 1974; Stroh, 1971). These systems have consistently shown considerable performance improvements over those which do not use adaptive quantizers. The adaptation procedures do require a mild increase in system complexity; some, however, can be realized with simple shift-and-add operations (Qureshi and Forney, 1975).

Adaptive quantizers have been the subject of intense theoretical study (Goodman and Gersho, 1974; Mitra, 1974a; Mitra, 1975; Cohn and Melsa, 1975). One result of significant intuitive interest unites two apparently different schools of thought on adaptation design. One approach, used in both backward and forward adaptation, normalizes the quantizer input by an estimate of its envelope or RMS level. The other method, first proposed by Jayant (1973), scales the normalization constant according to the prior output; if it is an outer level, the constant is

increased, if it is an inner level, the constant is decreased. Cohn and Melsa (1975a) showed that if some mild conditions are met, the Jayant method is also an envelope estimation.

Initial designs of Jayant quantizers proved to be very sensitive to channel errors. Later work (Cohn and Melsa, 1975a; Goodman and Wilkinson, 1975; Cohn and Melsa, 1975b; Qureshi and Forney, 1975) has shown ways of modifying the algorithm to eliminate this problem.

It is clear then, that adaptive quantizers improve performance because they match their range to the dynamic range of the incoming signal. A pitch compensating quantizer (Cohn and Melsa, 1976) extends this notion by noting that during voiced speech the dynamic range of the input is critically dependent on the proximity of the last pitch pulse.

It is well known that during voiced speech, the signal strength is a local maximum shortly after a pitch pulse and tends to decay towards the end of pitch period. This effect is even more pronounced in the difference signal that is quantized in DPCM systems. Although the predictor may closely approximate the actual speech away from a pitch pulse, it generally cannot predict the next pitch pulse. The design objective of a pitch compensating quantizer is to adapt to both the long term syllable variations in signal strength and to the short term variations.

The backward adapting pitch compensation algorithm uses a quantizer whose outermost levels have been set at higher values than is normal. When the quantizer output is one of these outermost levels, the adaptation algorithm reacts as if a pitch pulse had been detected; the quantizer state variable is significantly increased and then permitted to rapidly decay back to its long-term value.

The pitch compensating quantizer (PCQ) is employed in this algorithm. The prediction error e(k) is the input to the quantizer whose basic design is illustrated in Fig. 3.2. The recommended thresholds are symmetric and are illustrated in Fig. 3.3 and listed in Table 3.1. The level in which the normalized input falls specifies the quantizer output q(k). The inverse quantizer output e(k) is the quantized version of the quantizer input. It is the product of a scaling factor f(q(k)) and the state variable $\sigma(k)$. The recommended scale factors are tabulated in Table 3.2. The recommended thresholds were computed to be equidistant between between the scaling factors.

The state variable $\sigma(k)$ is designed to be an approximation to the standard deviation of e(k). Most of the time the scaled average of $|y(k)|$ is an acceptable estimate. However, in voiced speech at the beginning of a pitch period, e(k) is much larger than usual. Therefore, whenever one of the outermost quantizer level occurs, $\sigma(k)$ decays back to the scaled average of $|y(k)|$. Thus $\sigma(k)$ is updated by

$$\sigma(k) = \max\{\text{SMIN} <|y(k)|>, \phi[q(k)]\ \sigma(k-1)\} \qquad (3.1)$$

The first term in the braces of Eq. (3.1) usually dominates. This means that quantizer behavior is largely determined by SMIN$<|y(k)|>$ and, hence, by the product of SMIN and RMSMIN. It is recommended that the scale factor SMIN be set to 0.25. The second term in the braces only affects performance at the beginning of pitch periods. The quantizer expansion factors $\phi[q(k)]$ are given in Table 3.2.

The choice of output levels, expansion factors and other scalars in the design of quantizer is largely governed by maintaining the average

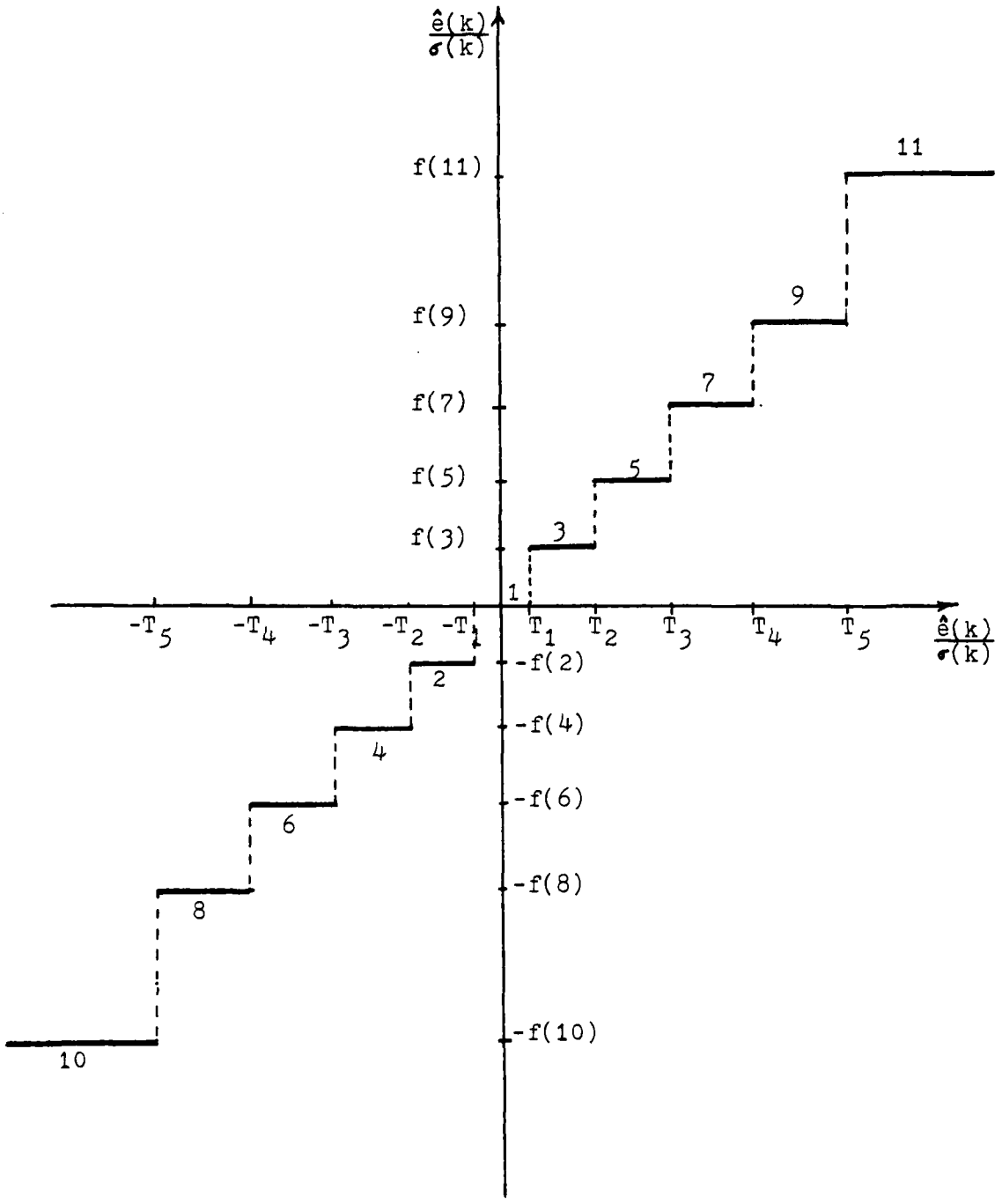


Fig. 3.3 An adaptive quantizer with thresholds and output levels.

TABLE 3.1

Quantizer Thresholds

i	T_i
1	0.3
2	1.05
3	2.55
4	5.55
5	11.55

TABLE 3.2

Quantizer scaling and expansion factors

q(k)	f[q(k)]	ϕ[q(k)]
1	0.0	0.8
2,3	0.6	0.9
4,5	1.5	9.0
6,7	3.0	1.5
8,9	6.0	3.0
10,11	12.00	6.0

source entropy within acceptable limits and at the same time reducing the quantization noise. Usually, the average number of bits available for coding quantizer levels is dependent on bit rate available and the sampling frequency. In this algorithm, fixed length code with run length is used. This requires that quantizer level occupancy statistics be such that probability of run lengths is maintained within a certain range. This makes the average bit rate remain below the allowable limit. The detail discussion may be found in Chapter 4.

The ARC system employs a sequentially adaptive linear predictor. It produces a linear prediction p(k) given by

$$p(k) = \sum_{i=1}^{N} a_i(k)\, \hat{y}(k-i) \qquad (3.2)$$

which is to be an estimate of y(k). The y(k-i) are the receiver's estimate of y(k-i). Since y(k) is frequency compressed to half the original bandwidth, the number of poles required to represent compressed bandwidth would be half that required for original bandwidth. A predictor order of four was found to be a good choice.

If the $a_i(k)$ accurately model the y(k), and if the y(k-i) are close to the y(k-i), then p(k) will be a good approximation to y(k). The $a_i(k)$ are adaptive, and after p(k) is formed, they are updated. They are adapted according to steepest descent of $e^2(k)$ [Melsa & Cohn, 1975]. This is approximated in the system by the following updating algorithm:

$$a_i(k+1) = \delta b_i + (1-\delta)\left[a_i(k) + \frac{g\, \hat{y}(k-i)e(k)}{\langle|\hat{y}(k)|\rangle^2}\right. \qquad (3.3)$$

where $\langle|y(k)|\rangle$ is a biased exponential time average of $|y(k)|$

$$\langle|\hat{y}(k)|\rangle = (1-\alpha)\sum_{j=0}^{\infty}\alpha^{j}|\hat{y}(k-j)| + \text{RMSMIN}$$

Thus, the $a_i(k)$ updating algorithm has eight parameters: δ,g,α,RMSMIN and b_i for i=1,2,3 and 4. It was found in the simulation studies that the effect of channel errors become more severe if the memory in the updating process is increased. This increase is essentially controlled by choice of parameters δ,q and α. In order to minimize the effect of channel errors, the memory time was reduced from what would be optimal in error-free case. This did not significantly degrade performance. The recommended values of these parameters are

$$\delta = 0.05$$
$$g = 0.02$$
$$\alpha = 0.93$$

The parameters b_i represent the quiescent values of the coefficients $a_i(k)$. The values used are

$$b_i = \begin{cases} 0.7 & i=1 \\ 0 & i=2,3 \text{ or } 4 \end{cases}$$

The quantity RMSMIN is perhaps the most sensitive parameter in the algorithm. It determines the minimum value of $\langle|y(k)|\rangle$ which affects both the adaptive predictor and the adaptive quantizer. As the RMSMIN decreases, the system responds more during low level signals. This reduces granular noise and increases the data rate. The higher data rate means that the buffer fills faster and that buffer control is triggered causing deterioration of speech quality. When y(k) is represented on the interval [-2048, 2047], RMSMIN of 40 produces a good tradeoff.

3.3 PERFORMANCE

The performance of the TDHS-ARC system depends on how well the frequency scaling operations are performed and how well the quantization is done. As mentioned earlier, the ARC system is essentially an ADPCM system and for the given bit rate, its performance can be improved by decreasing the quantization noise. However, a reduction in the quantization noise does not necessarily mean a improvement in the performance. Such situations will be discussed later on in this section.

The bit rate available for coding quantizer levels depends on the bit rate needed for error protection and to transmit the side information, if any. Since it was decided to extract pitch at the transmitter and at the receiver as well, this system has no side information. With (26, 31) Hamming code for error protection [Hamming, 1980], a bit rate of 8 kbs is available to code the quantizer levels generated by ARC. The sampling rate is 3200 Hz, which leaves an average of 2.5 bits per sample. The parameters were designed such that the average value of the source entropy is around 2.5 bits per sample. However, the instantaneous value of the bit rate has to be considered for the buffer overflow problem.

Before discussing the performance of the ARC system, some of the performance indicators are presented. The single most widely used indicator of speech coder performance is the SNR defined by

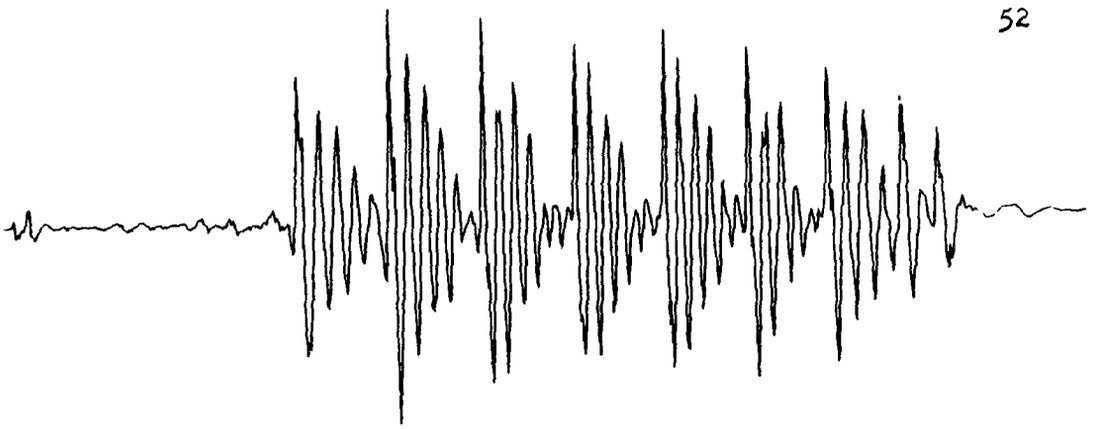
$$\text{SNR} = \frac{\langle y^2(k) \rangle}{\langle |y(k)-\hat{y}(k)|^2 \rangle} \tag{3.4}$$

or in dB by

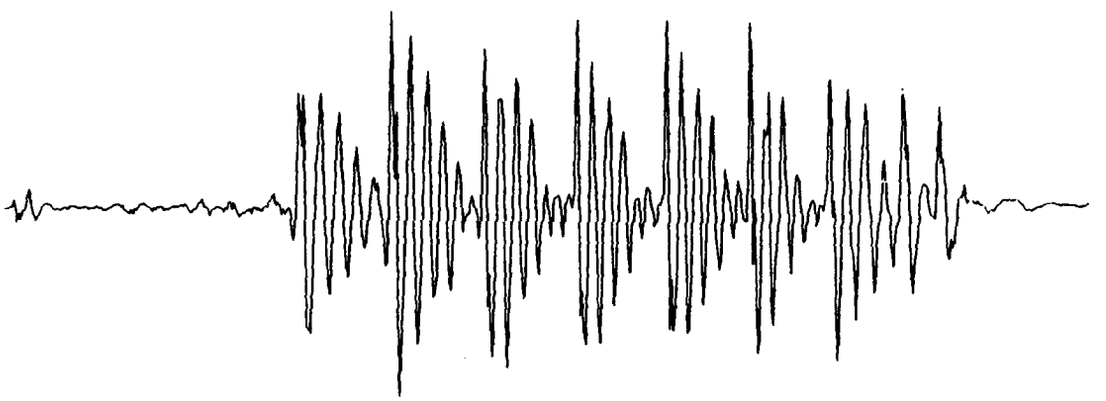
$$SNR(dB) = 10 \log_{10} SNR \qquad (3.5)$$

where y(k) and y(k) are shown in Fig. 3.4(a) and (b). $\langle|\,|^2\rangle$ denotes the averaging operation. It is well known that the SNR calculated as in Eqs. (3.4) and (3.5) is not a perfect indicator of speech quality and intelligibility. This is because perceived quality and intelligibility depend on the subjective loudness of the quantization noise not just on the quantization noise power, which is the denominator of Eq. (3.4). The numerator of Eq. (3.4) is an average of a square term. Thus, speech with high amplitudes gets more weighing in the calculation of the SNR. Makhoul & Berouti (Feb. 1979) and Atal & Schroeder (June 1979) have shown that due to the auditory masking properties of the human ear, the subjective loudness is determined by the spectrum of the quantization noise and its relationship to the input signal spectrum. The SNR in (3.4) does not reflect these considerations. However, the SNR remains popular since it is simple to compute, and it can be useful if applied prudently.

Usually, SNR is calculated for the entire utterence. As discussed earlier, the high energy segments of speech get more weighing in the SNR computation than the low energy segments. Hence, SNR values indicate the performance of coder for high energy speech, or voiced speech. The coder performance for low energy, or unvoiced speech can be obtained by computing the SNR in Eqs. (3.4) and (3.5) over many nonoverlapping blocks of data within an utterence [Noll, 1975]. The time variation of the resulting "short-term" SNR provides an indication of how well the coder under consideration is performing on the various blocks of speech data. Based on these short term SNR values, a performance measure SNRSEG can be defined as



(a)



(b)

(c)

Fig. 3.4 (a) Compressed speech, y(k)
(b) Reconstructed compressed speech, $\hat{y}(k)$
(c) Quantization error, e(k)

$$\text{SNRSEG} = \frac{1}{k} \sum_{j=1}^{k} \text{SNR}_j(\text{dB}) \tag{3.6}$$

Infrequent very large values of SNR_j tend to show up better in SNRSEG than in SNR in (3.4) [Jayant, 1977]. One slight modification to computation of SNRSEG in (3.6) would be to exclude the SNR_j values for the blocks which contain silence. Usually, SNR values for silence are zero or negative; those can be excluded from SNRSEG computation.

A distortion measure that is related to the SNR is the SPER given by [Gibson, 1980]

$$\text{SPER} = \frac{\langle y^2(k) \rangle}{\langle [y(k) - p(k)]^2 \rangle} \tag{3.7}$$

where p(k) is the predicted value in Fig. 3.1. The SPER is motivated by a desire to evaluate the predictor performance only, rather than to try to obtain an indicator of the overall system performance which in DPCM includes both the quantizer and the predictor effects. Of course, due to the closed loop nature of DPCM and the presence of the quantizer within the loop, the SPER is also affected by the quantizer.

Another performance indicator for DPCM is the SNRI defined by [McDonald, 1966; O'Neal & Stroh, 1972]

$$\text{SNRI} = \frac{\langle y^2(k) \rangle}{\langle [y(k) - p_1(k)]^2 \rangle} \tag{3.8}$$

where

$$p_1(k) = \sum_{i=1}^{N} a_i y(k-i).$$

The SNRI is simply the SPER when it is assumed that $y(k) \approx y(k)$ or $p(k) \approx p_1(k)$. The SNRI is interpreted as the amount by which linear prediction

can reduce the input signal power and hence is sometimes used as a measure of maximum utility of linear prediction in DPCM.

Although the SNR, SPER and SNRI are easy to calculate and hence very useful for initial system evaluations, they are not absolute indicators of system performance. In fact, the SNR may not rank the coders correctly in terms of speech quality and intelligibility. As a result, it is advisable to augment these previous indicators with subjective listening tests and frequency spectrum plots. By comparing frequency spectrum of the speech coder output with frequency spectrum of the original speech, conclusions can be drawn concerning how well the coder tracks the various formants, reproduces unvoiced or voiced sounds and in frequency quantizer noise in prevalent. By combining all these techniques, coder degradation can be analyzed.

For evaluating the performance of the ARC and also of the entire system, the following three phonetically balanced sentences were used.

1) "Cats and dogs each hate the other" (Male speaker).

2) "Move the vat over the hot fire" (Male speaker).

3) "The pipe began to rust while new" (Female speaker).

The input data were low pass filtered at 2900 Hz (3dB) and sampled at 6.4 kHz. The SNR and the SEGSNR in time and frequency domain are given in Table 3.3. Frequency domain SNR indicates how good the match between input and output speech in frequency domain is. It is calculated by taking DFT of small segment (20 msec) of input and output speech and calculating SNR, as in Eq. (3.9)

TABLE 3.3

ARC Performance

Block length = 128 samples, Sampling freq. = 3200 Hz

Sent #	SNR	SEGSNR time	SEGSNR frequency	Entropy H
Male 1	20.00 dB	16.39 dB	19.28 dB	2.47 b/sample
Female 11	19.83 dB	17.57 dB	19.74 dB	2.59 b/sample
Male 6	20.01 dB	18.12 dB	20.61 dB	2.57 b/sample

$$SNR_{freq} = 10 \log_{10} \frac{\langle |Y(n)|^2 \rangle}{\langle [|Y(n)| - |\hat{Y}(n)|]^2 \rangle} \qquad (3.9)$$

where Y(n) and Y(n) are frequency components of y(k) and y(k). Fig. 3.4 shows the compressed speech y(k), reconstructed compressed speech y(k), and the quantization error. It can be seen that the ARC system reconstructs the input speech very well, especially in the voiced speech segment. This is more clear in SEGSNR plot of Fig. 3.5, where block to block SNR is plotted for the whole utterence against time. The plot shown in the dashed lines is for the SNR in frequency domain. The distortions in frequency domain can be seen from the frequency spectrum plots of input and output speech as well as input and quantization noise in Figs. 3.6(a) and (b). The frequency spectrum of quantization noise lies much below that of input compressed speech, near the formant frequencies especially for first and second formant. At higher frequencies, however, quantization noise is perceptable since its spectrum lies above the input signal spectrum. In informal listening tests, it was found that this granular noise problem is much less than that in previously developed ADPCM or APC system employing waveform reconstruction techniques. Noise spectral shaping was tried by a number of authors [Atal & Schroder, 1979; Makhoul & Berouti, 1979] to improve the speech quality. However, such spectral shaping adds to the complexity of the system. In the previous research studies for developing PARC system [Melsa, et al., 1980], it was noticed that increases in complexity and levels of adaptation in the system leads to poorer performance in the presence of channel errors. Hence, the compromise between the robustness and speech quality

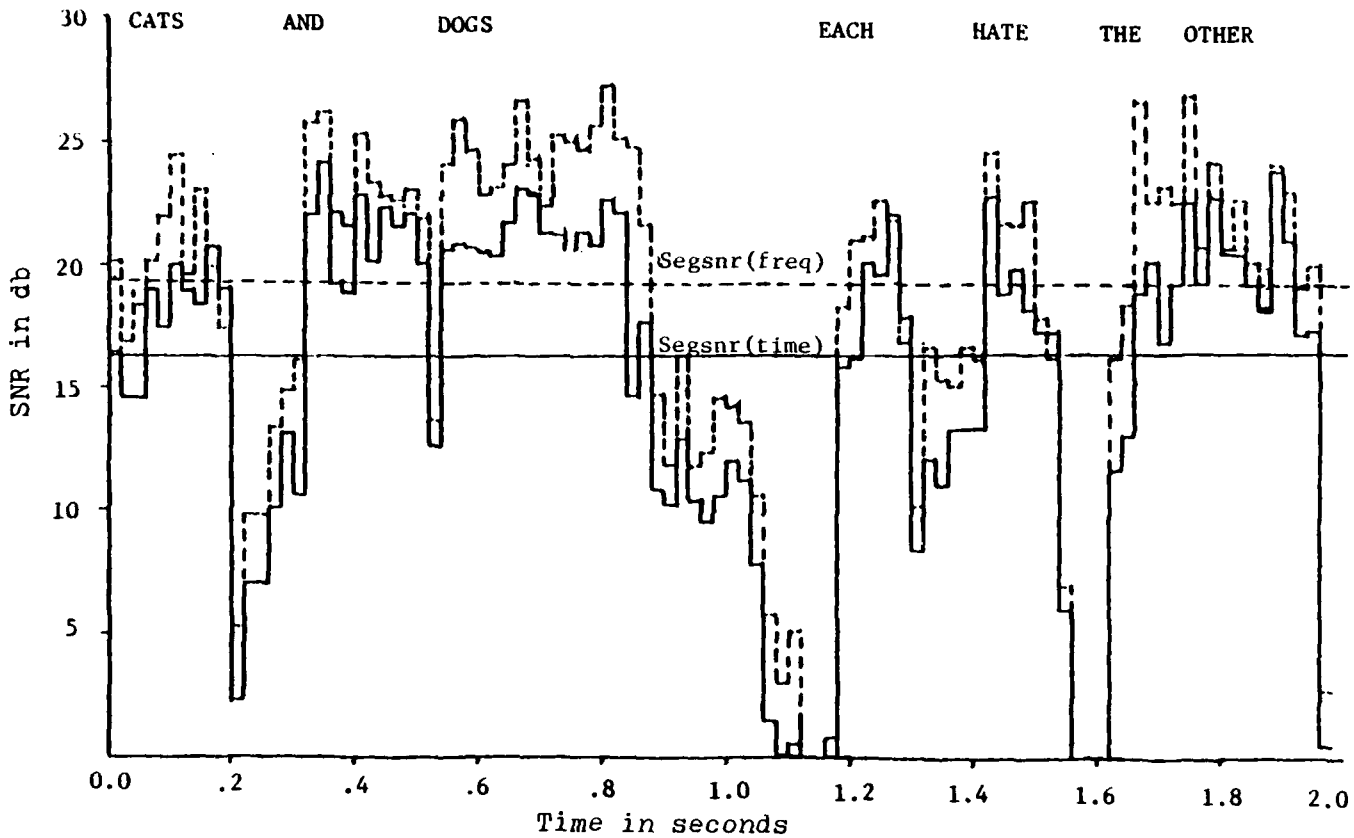


Fig. 3.5 Block-to-block SNR plot for male utterance in ARC system.

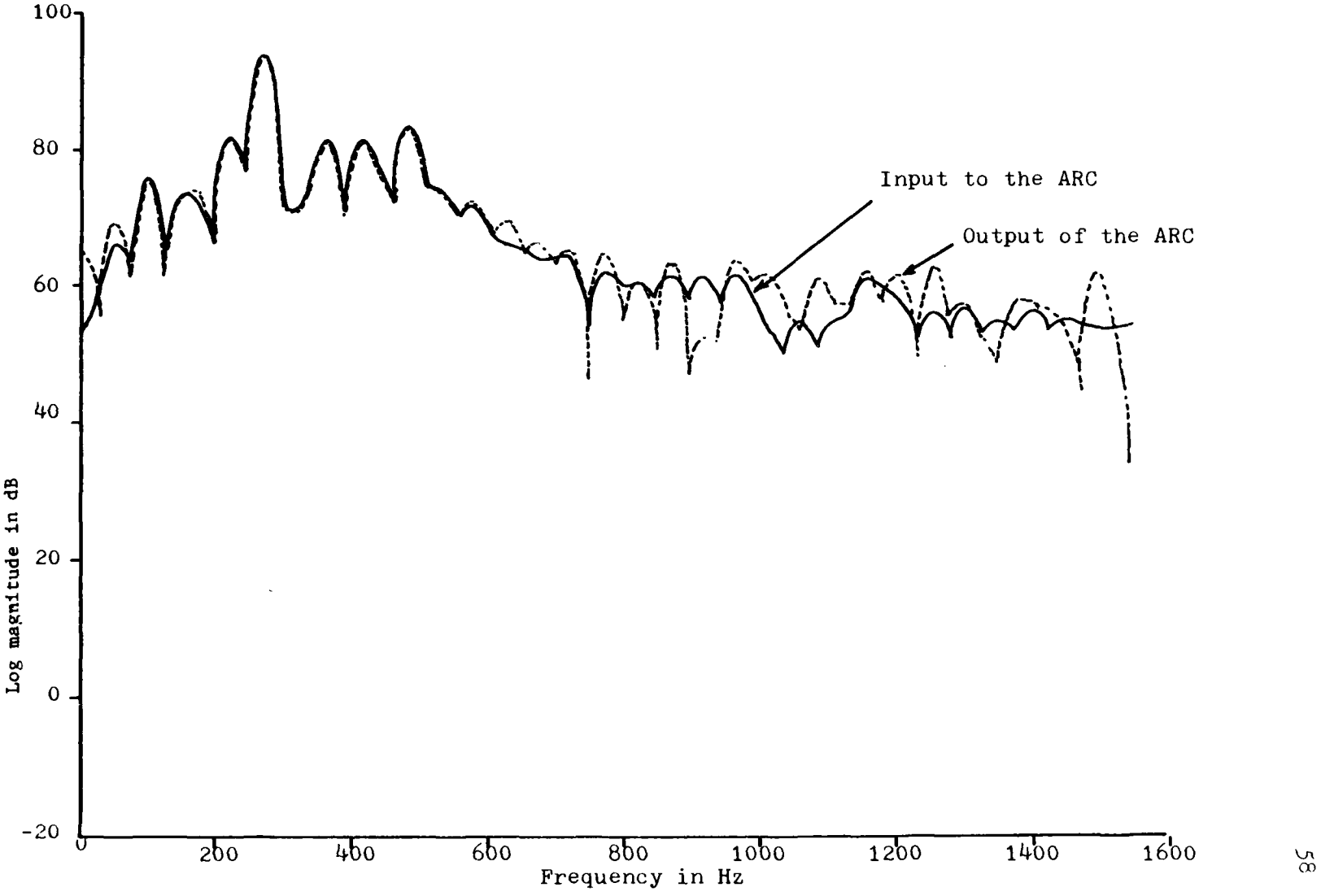


Fig. 3.6(a) Frequency spectrum of input and output speech of Adaptive Residual Coder.

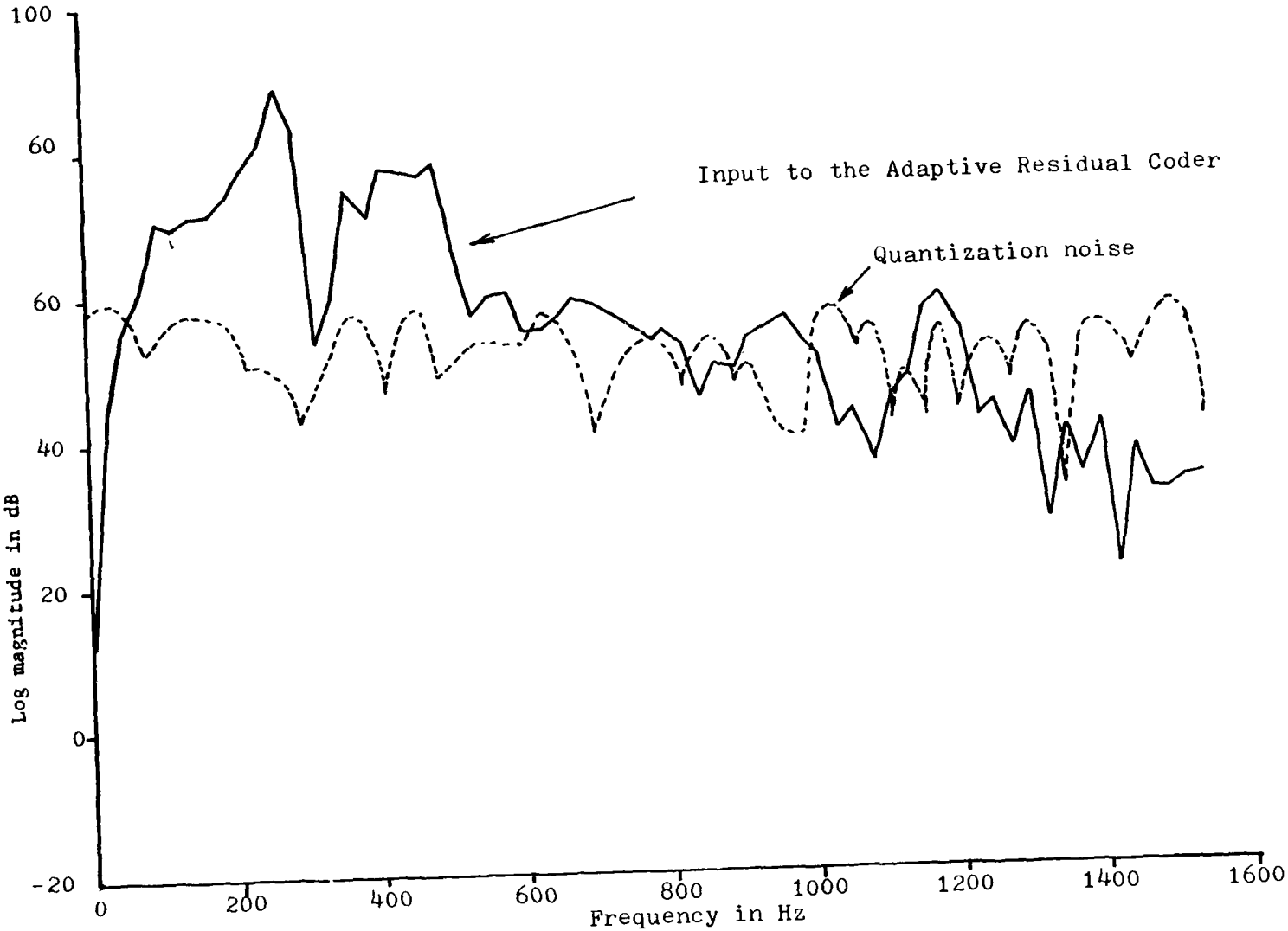


Fig. 3.6(b) Frequency spectrum of compressed speech and quantization noise.

must be reached in the design of the ARC system. The Fig. 3.7 show the plot of SNR against the signal strength. The desirable performance would be a high value of SNR for all signal strengths. Such performance is unlikely since the predictor performs very poorly for unvoiced speech which is noiselike. However, the parameters were designed such that ARC performance approaches such curve.

The quantizer and the predictor performance for segment to segment of speech is shown in Figs. 3.8(a) and (b). The performance indicator, SNRI in Eq. (3.8), is also plotted in Fig. 3.8(a). It can be seen that the predictor performance, SPER is very close to SNRI. That means that the predictor is performing very well except in transition regions, where its poor performance is anticipated. Total performance (SNR in dB) is the addition of predictor performance (SPER in dB) and the quantizer performance (SNRQ in dB). Hence, increases in SPER and SNRQ lead to an increase in SNR. However, such simplification could be misleading since performance of the predictor depends on that of quantizer. The plots, in Figs. 3.8 often are useful in evaluating the contribution of the predictor and the quantizer in different parts of the speech utterence.

3.4 SUMMARY

The design of adaptive residual coder for coding the compressed speech was presented. An 11-level quantizer and 4th order predictor was used. The signal to quantization noise ratio as high as 20 dB could be obtained. Various performance measures to indicate the speech quality, were presented. It was found that no single criterion indicates the true speech quality. The study of combined system using TDHS and ARC is presented in the next chapter.

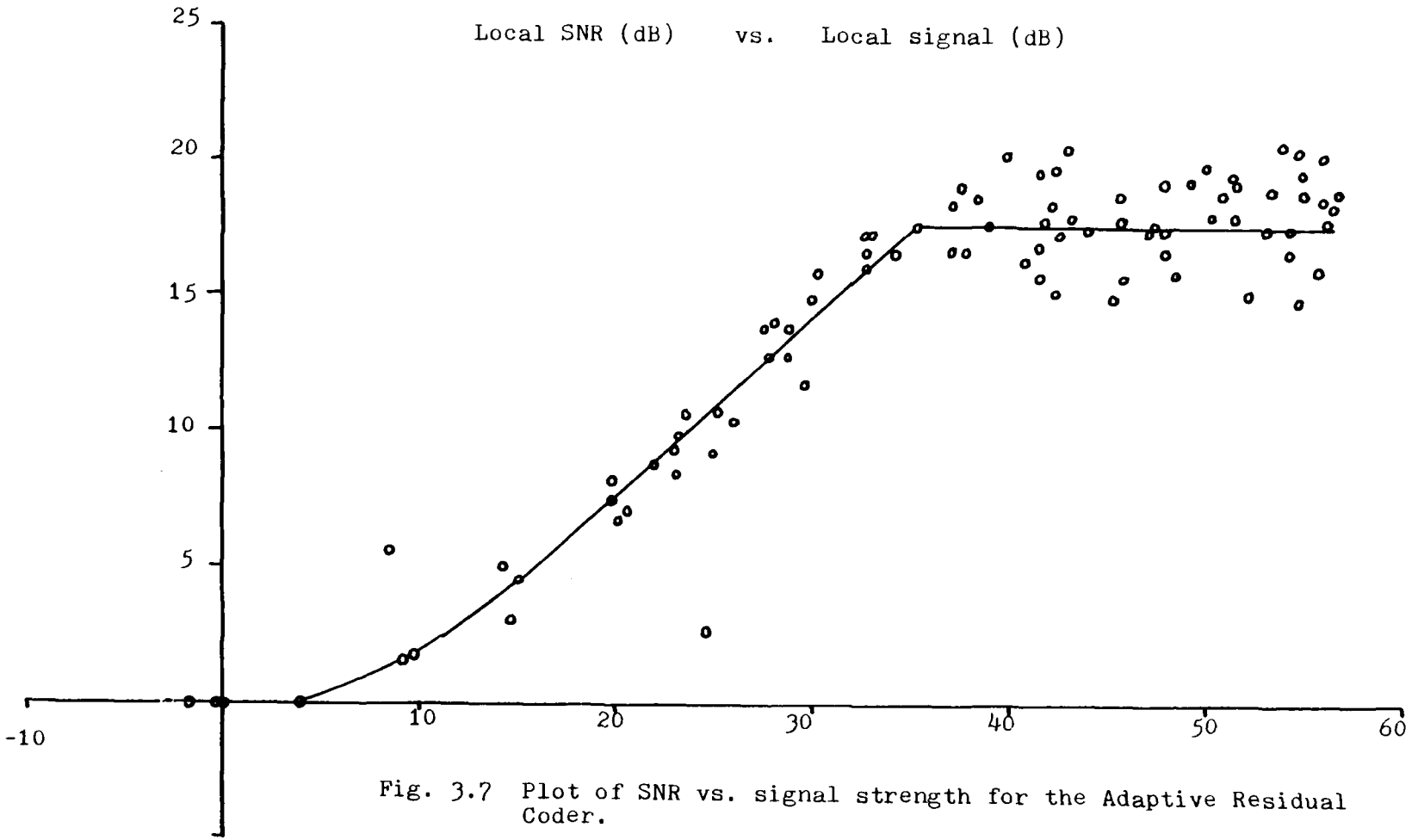


Fig. 3.7 Plot of SNR vs. signal strength for the Adaptive Residual Coder.

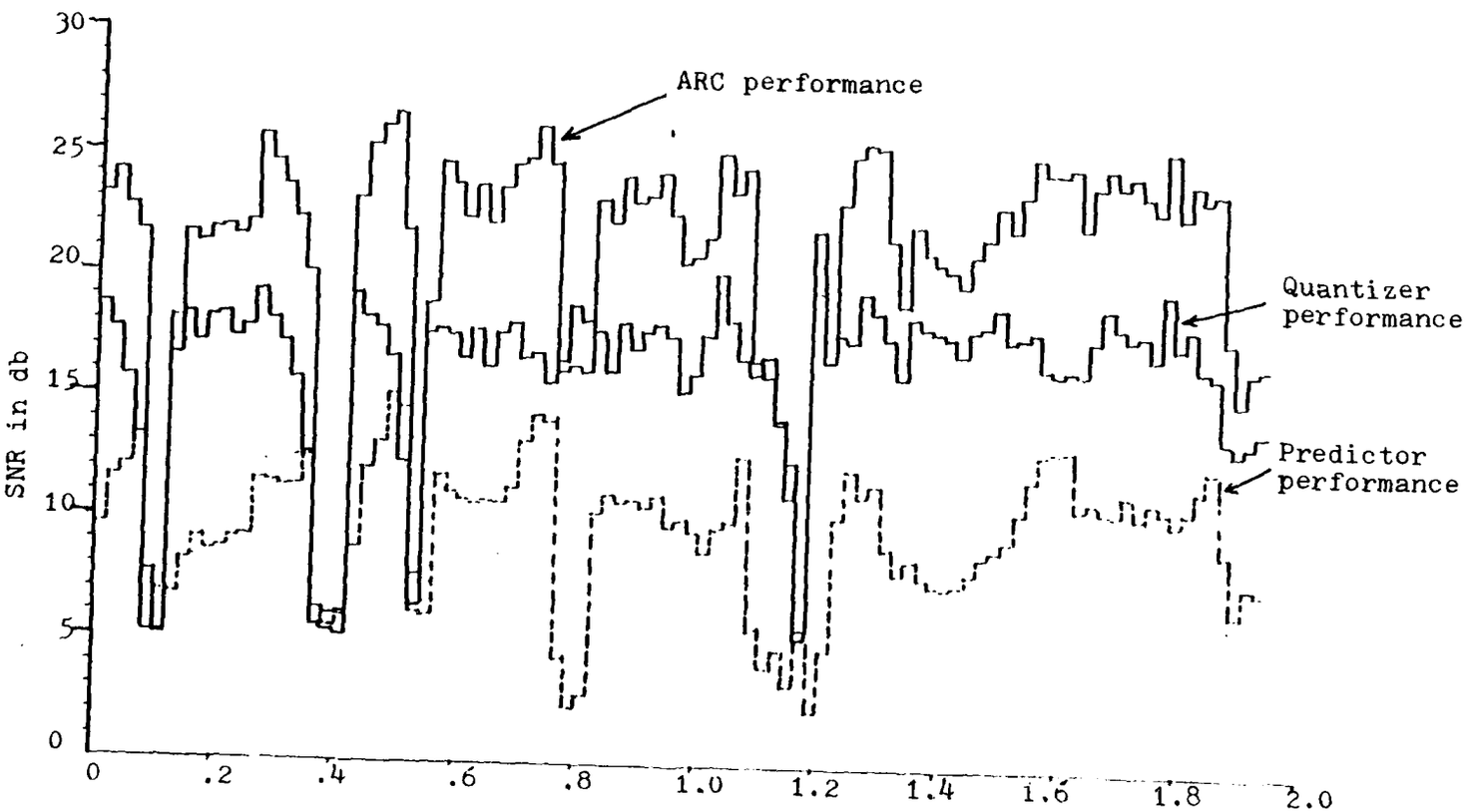


Fig. 3.8(b) Block-to-block SNR, SNRQ and SPER for female speaker.

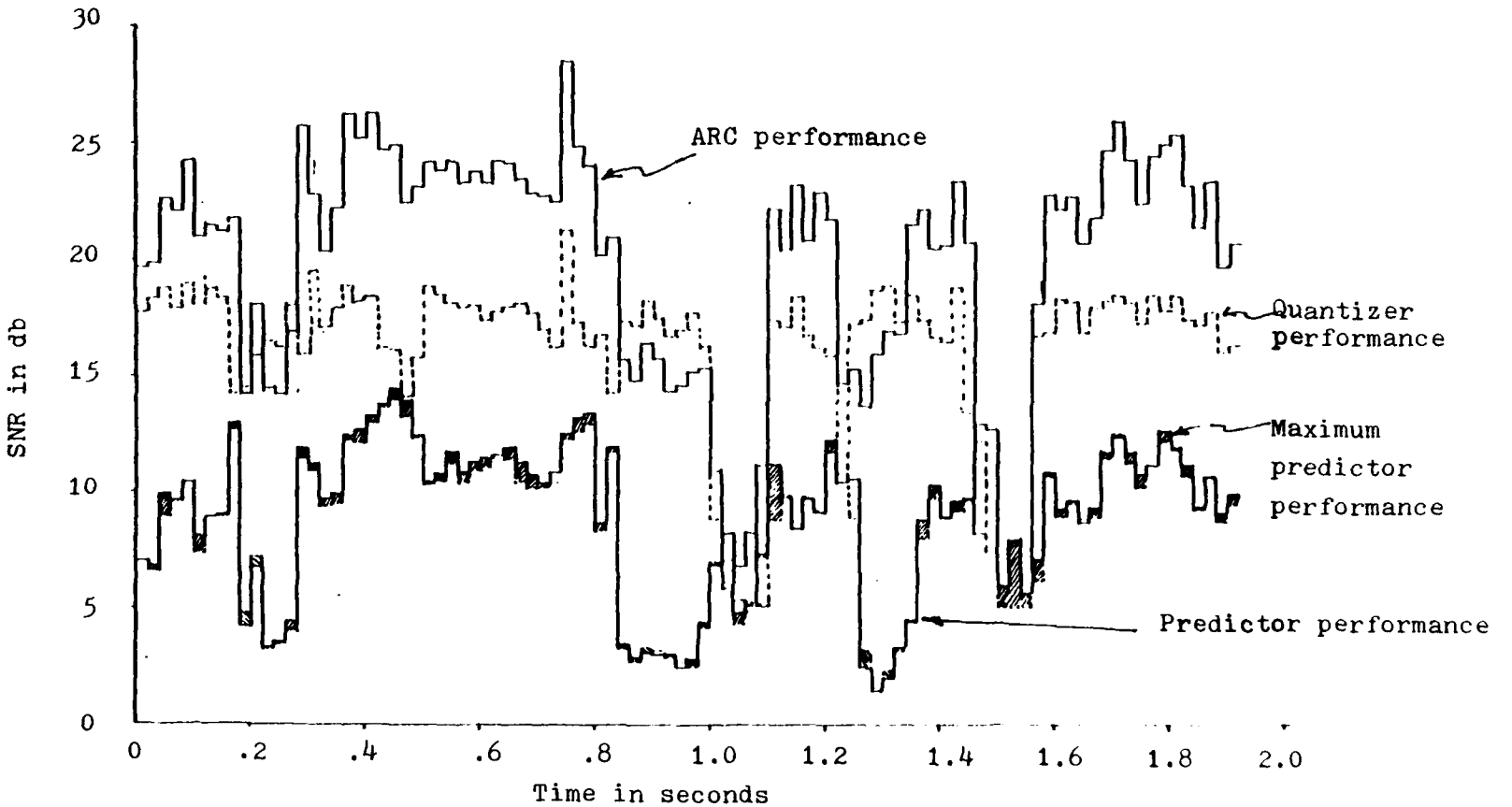


Fig. 3.8(a) Block-to-block SNR, SNRQ, SPER and SNRI for male speaker.

CHAPTER 4

TIME DOMAIN HARMONIC SCALING (TDHS) and ADAPTIVE RESIDUAL CODER (ARC) SYSTEM

4.1 INTRODUCTION

This chapter describes the complete system employing harmonic scaling and residual coding operations to achieve the desired transmission bit rate. As discussed earlier, time domain harmonic compression reduces the number of samples to be transmitted, and the adaptive residual coder encodes these samples with least possible distortion. The complete system performance depends not just on the TDHS and ARC performance but also on several factors, such as the source code, the error protection employed as well as buffer control strategy. These and other topics were investigated and the results are presented here.

Section 4.2 describes two system configurations; one with pitch as side information and the other with no side information. The source code and the study of the buffer behavior is given in Sections 4.3 and 4.4 respectively. The effects of various transmission bit error rates are examined and reported in Section 4.5. The system performance for background noise is discussed in Section 4.6.

4.2 SYSTEM CONFIGURATION

Fig. 4.1 shows the system structure of TDHS-ARC system. For simplicity the A/D and D/A converters and associated filters are not shown. The speech samples, s(k), bandlimited at 3.2 kHz and taken at Nyquist frequency, form the input. The pitch period, N_p, is extracted from a

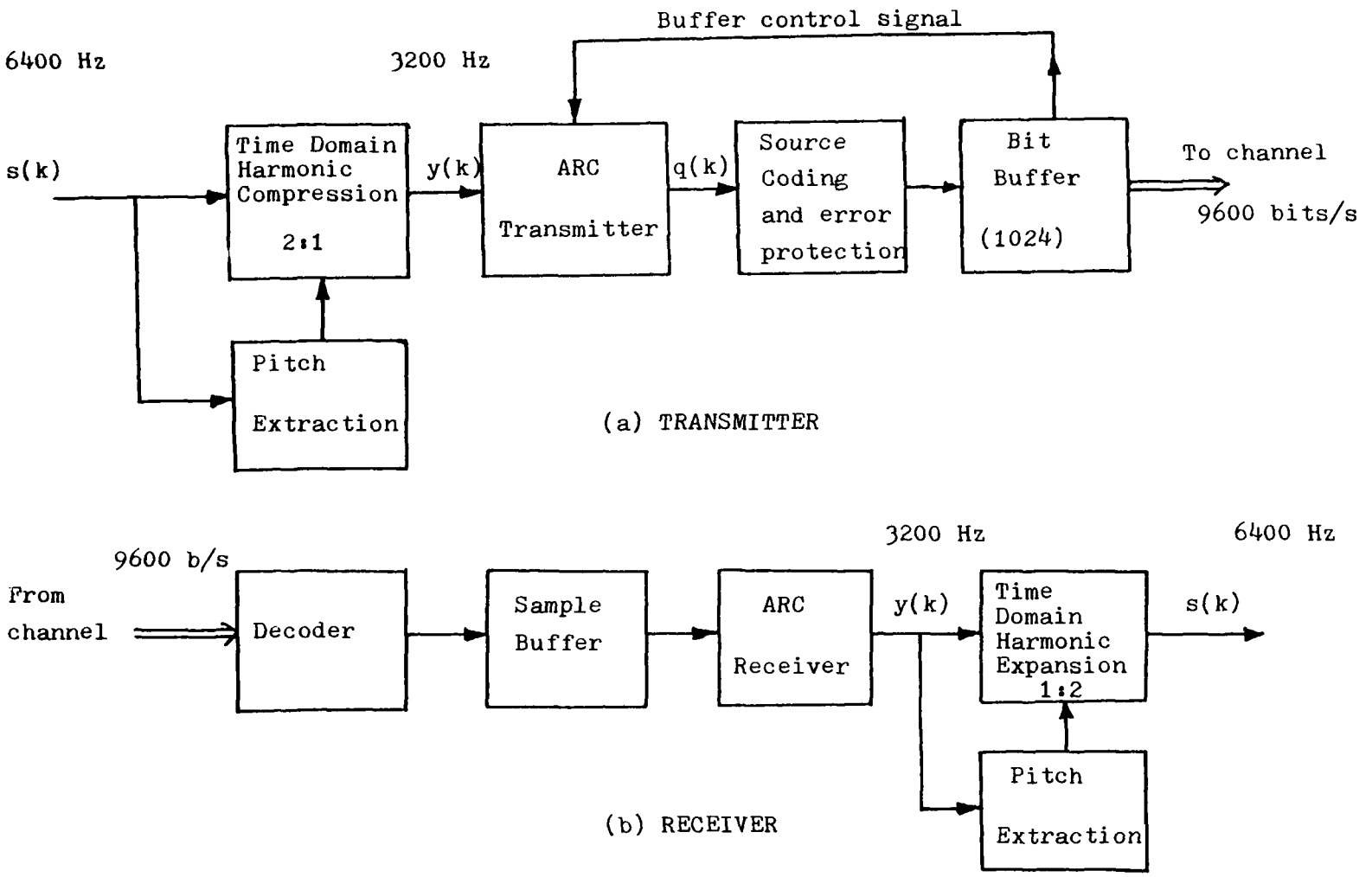


Fig. 4.1 TDHS-ARC system.

block of speech samples using the center clipped 3 value autocorrelation method. This value of the pitch period is transferred to the time domain harmonic compression algorithm. This algorithm produces N_p samples of compressed speech y(k), as described in Chapter II. The sampling frequency now becomes half of the original Nyquist frequency. The smoothly varying compressed speech samples are coded using an Adaptive Residual Coder. The output of ARC is a string of quantizer outputs q(k). For every sample or two samples (if two samples form the desired runlength), a 4-bit word is transmitted to the bit buffer. For the bit rate of 9.6 Kb/s and the sampling frequency of 3.2 kHz, the average number of bits per sample transmitted on the channel is 3. Depending on the Hamming code chosen for the error protection, the parity bits are also added to the buffer.

The length of the buffer was chosen to be 1024 bits which corresponds to approximately a tenth of a second delay. Depending on the bit rate generation, the buffer may overflow or underflow. To avoid such drastic situations, buffer content information is passed on to the ARC transmitter to take some action to control the buffer. This is explained in Sec. 4.3. Note that the bits transmitted on the noisy channel represent only the quantizer level information.

Bits thus transmitted are decoded at the receiver correcting any occurrence of single channel error in a Hamming block. The received quantizer level q(k) forms the input to the ARC receiver whose output is the reconstructed compressed speech y(k). The time domain harmonic expansion is performed on these samples. However, the pitch information is needed for the frequency multiplication operation. There are two ways to pass on the pitch

periods to the harmonic expansion operation. One scheme is to extract the pitch periods at the transmitter and send them to the receiver as side information. However, such scheme is associated with a number of disadvantages. The biggest disadvantage is the need for a blocking structure to transmit the quantizer levels. Every $2N_p$ samples are associated with the pitch period N_p. It is very important that every block of samples at the receiver must be associated with the same pitch period as that at the transmitter. This synchronization can easily be lost unless special error protection is provided for pitch information. However, such error protection as well as pitch period transmission would cost considerable bit rate. This would leave fewer bits per sample for transmitting quantizer levels and therefore, would cause more quantization noise.

In the case of pitch transmission it has been seen that the matching of pitch data and the block of quantizer levels, which is itself pitch dependent, is very sensitive to the transmission errors. Besides, for the 9.6 kb/s system, the extra bit rate required for error protection of the pitch information can not be afforded. This led to the implementation of the other scheme with pitch extraction at the receiver (see Fig. 4.1). Since the pitch period is estimated from the reconstructed compressed speech rather than reconstructed original speech, there are fewer pitch periods per unit time available for pitch extraction. This problem can be handled by using smaller blocksize and the searching range. Table 4.1 shows the pitch extracted at the transmitter and at the receiver. Since the pitch at both ends is extracted from a different type of signal as well as different number of samples, comparison should be made for the

TABLE 4.1

Comparision of pitch periods extracted at the transmitter and at the receiver.

At the Transmitter		At the Receiver		At the Transmitter		At the Receiver	
Sample #	Pitch	Sample #	Pitch	Sample #	Pitch	Sample #	Pitch
1099 - 1298	27	545 - 744	27	5333 - 5532	29	2664 - 2863	29
1153 - 1352	82	572 - 771	55	5391 - 5598	38	2693 - 2892	38
1317 - 1516	27	627 - 826	55	5451 - 5650	38	2723 - 2922	38
1371 - 1578	27	682 - 881	55	5511 - 5718	38	2753 - 2952	38
1425 - 1624	82	737 - 936	55	5571 - 5778	38	2783 - 2982	38
1589 - 1788	55	792 - 991	28	5631 - 5838	38	2813 - 3012	38
1699 - 1898	55	820 - 1019	28	5691 - 5898	38	2843 - 3042	38
1809 - 2008	27	848 - 1047	27	5751 - 5958	38	2873 - 3072	38
1863 - 2062	27	875 - 1074	27	5811 - 6018	38	2903 - 3102	38
1917 - 2116	28	902 - 1101	27	5871 - 6078	38	2933 - 3132	38
1973 - 2172	28	929 - 1128	27	5931 - 6138	38	2963 - 3162	38
2029 - 2228	28	956 - 1155	28	5991 - 6198	38	2993 - 3192	38
2085 - 2284	28	984 - 1183	28	6051 - 6258	38	3023 - 3222	38
		1012 - 1211	29	6111 - 6318	38	3053 - 3252	38
		1041 - 1240	28	6171 - 6378	38	3083 - 3282	38
				6231 - 6438	38	3113 - 3312	38
				6291 - 6498	38	3143 - 3342	38
				6351 - 6558	38	3173 - 3372	38
				6411 - 6618	38	3203 - 3402	38
				6471 - 6678	38	3233 - 3432	38

voiced and unvoiced speech segment. Remember that the number of samples in any particular voiced segment of compressed speech is half as many as that in a voiced segment of original speech. The pitch estimator does surprisingly well in voiced regions of the utterence. However, in the transition regions between voiced and unvoiced speech pitch estimation at the receiver differs significantly from that at the transmitter. A similar situation occurs at the boundary of the two different sounds. The distortion caused by the above situations is audible in form of buzziness which can be heard only through very high quality headphone. It was thought that such distortion could be tolerated to preserve the robustness and the simplicity of the system.

Fig. 4.2 show the frequency spectrum of a 20 msec segment of input and output speech with and without pitch transmission. The speech is not quantized in order to focus attention on the distortion caused by frequency compression and expansion operation.

4.3 SIMULATION

A program was written in FORTRAN IV to simulate the Time Domain Harmonic Scaling, Adaptive Residual Coding, Coder, Decoder and Buffer control operations. The structure of the simulation is shown in Fig. 4.3. The transmitter program (TRAN·FTN) consists of the simulation of harmonic compression, pitch extraction and the adaptive residual transmitter program. This module produces the quantizer data (QUANT·DAT), pitch data (PITCH·DAT), and the output data (FOR006·DAT) files. The output data file has the record of parameters used, quantizer statistics, predictor quantizer and the ARC transmitter performance. The encoder program reads the

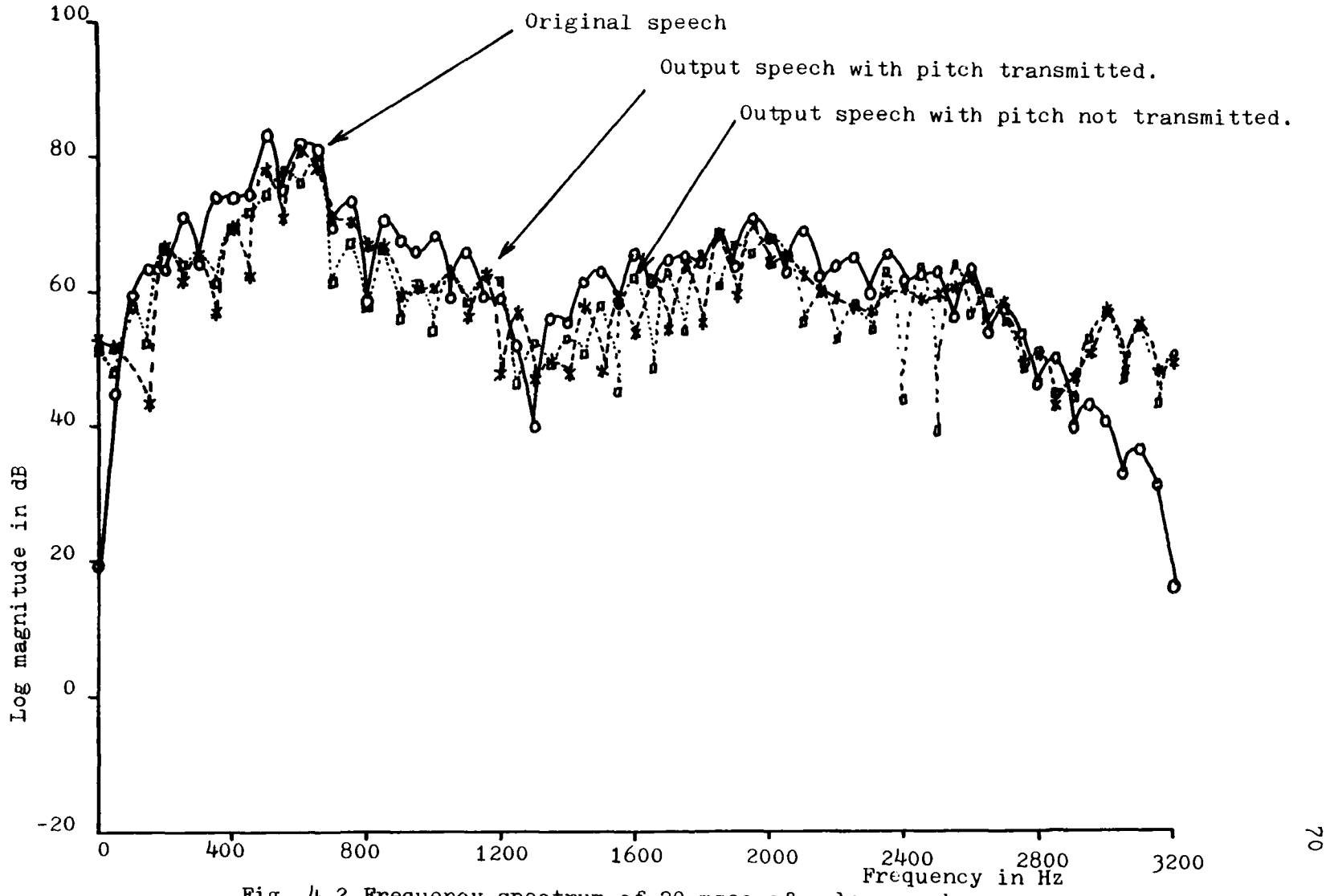


Fig. 4.2 Frequency spectrum of 20 msec of male speech.

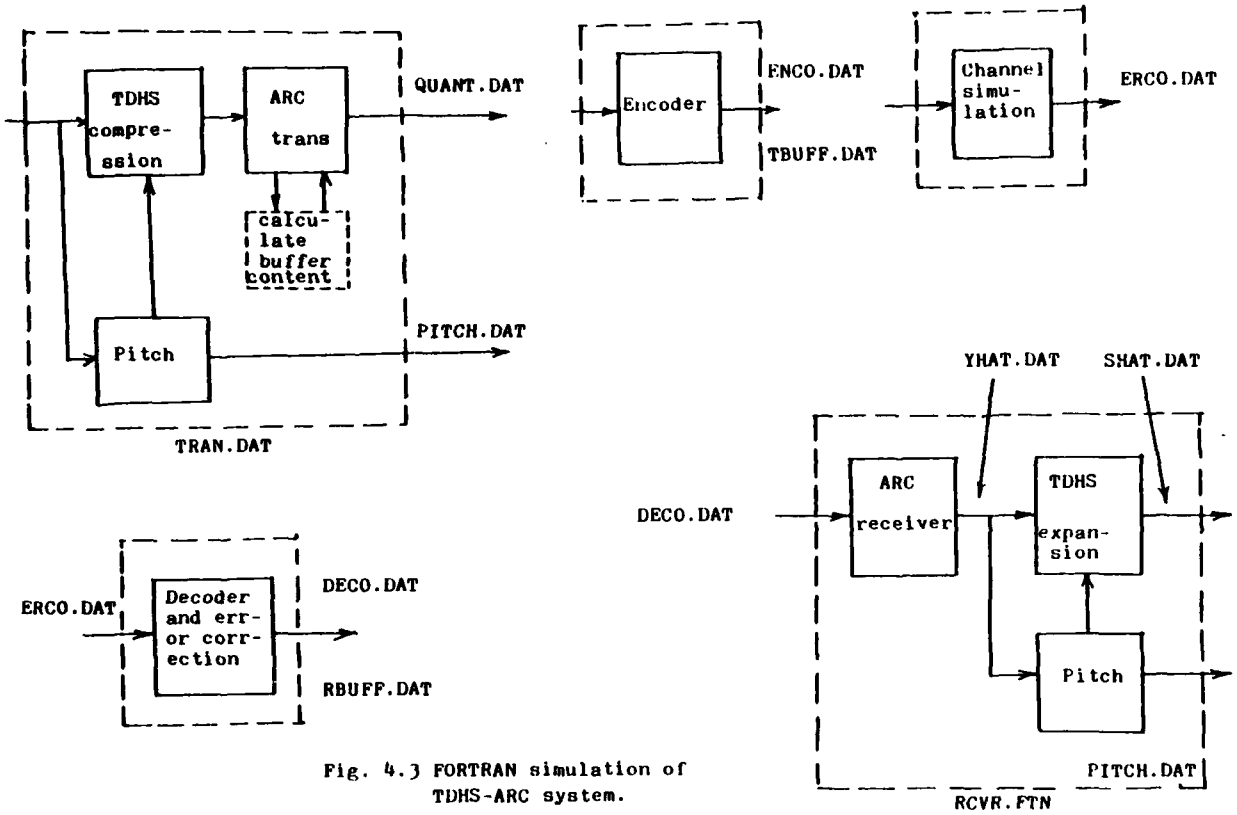


Fig. 4.3 FORTRAN simulation of TDHS-ARC system.

quantizer data from QUANT·DAT file and codes it and writes the code words in ENCO·DAT file. The encoder program also produces the transmitter buffer simulation file (TBUFF·DAT). The channel simulation program reads the code words from file ENCO·DAT and adds the desired channel errors in the bits and writes these corrupted code words in a file, called ERCO·DAT. The decoder program reads this file, decodes the code word into quantizer levels and produces receiver buffer simulation file (RBUFF·DAT) and the decoded quantizer level file (DECO·DAT). The receiver program combines the ARC receiver operation, pitch extraction and the time domain harmonic expansion operation. The reconstructed speech file (SHAT·DAT) and pitch data file (PITCH·DAT;2) is produced for comparisons with those at the transmitter.

Various simulation runs were made for the three utterences described in an earlier chapter. The combined system performance was evaluated by using the criteria outlined in Chapter 3. It was noticed that none of the objective measures described there indicate the true quality of output speech particularly in the case when pitch is extracted at the receiver. Table 4.2 shows the sliding SNR in the time and frequency domain for three utterances, two male and one female speaker. The negative value of SNR does not indicate the bad quality of output speech but it represents the phase difference between input and output speech. Most of the decisions regarding the speech quality were based on the informal listening tests. Fig. 4.4 show that the time plots of input and output speech for the segments of two utterences. It can be seen that the shape of the input speech waveforms is preserved.

TABLE 4.2

Sliding SNR in frequency domain for two male and one female two second utterence.

sentence 1 male speaker (dB)	sentence 6 male speaker (dB)	sentence 11 female speaker (dB)	sentence 1 male speaker (dB)	sentence 6 male speaker (dB)	sentence 11 female speaker (dB)
4.884	8.119	1.822	4.186	3.888	22.394
4.879	9.617	3.287	2.710	2.792	20.364
7.732	10.969	5.168	8.815	8.986	14.676
10.616	16.326	-1.163	6.892	9.862	19.662
12.633	4.426	0.000	3.780	6.050	22.027
15.020	10.370	0.000	3.943	3.023	19.660
7.721	10.287	3.997	4.426	5.709	20.119
8.318	13.726	6.170	1.232	8.415	14.637
8.996	8.650	8.613	0.000	14.430	10.969
1.193	12.123	16.776	0.000	16.909	4.272
1.620	16.043	16.393	0.000	16.040	3.785
10.070	4.606	16.631	8.093	10.819	5.500
5.039	10.932	21.246	3.326	-8.960	3.132
5.963	7.921	22.461	14.446	4.791	1.116
3.942	10.711	19.217	13.007	-8.324	8.612
4.161	11.260	17.144	11.300	5.942	2.996
13.012	10.002	12.047	8.443	7.811	7.533
11.933	5.429	8.133	3.737	6.445	19.127
12.572	12.200	0.000	1.074	6.496	17.490
12.200	13.703	0.000	2.005	4.036	17.304
15.990	16.294	0.000	6.127	-4.135	7.473
10.547	12.206	3.037	3.762	6.500	18.639
10.396	14.216	16.920	5.932	14.590	20.110
10.604	16.769	22.949	6.999	7.214	20.705
13.223	14.901	8.100	10.000	8.490	19.092
12.506	9.329	6.460	7.619	7.732	16.793
7.426	9.149	3.716	12.470	6.499	17.691
10.363	6.342	2.110	6.291	7.260	18.230
17.510	7.426	0.120	6.699	6.967	18.230
13.051	11.927	16.393	5.649	3.926	16.629
12.423	-2.100	20.790	3.260	7.266	16.771
11.617	7.120	22.720	0.000	7.310	16.106
20.020	13.330	22.260	0.000	4.699	17.410
10.232	16.745	20.976	0.000	3.701	21.665
12.323	14.420	18.703	7.524	4.976	22.037
10.090	21.059	20.426	7.169	5.000	16.006
12.479	18.199	22.900	9.503	-13.440	15.352
13.323	18.076	16.305	12.461	0.000	20.374
10.505	16.461	2.593	12.259	0.000	21.605
0.296	17.957	6.157	6.066	0.000	19.261
13.922	20.664	7.006	16.137	0.000	21.360
14.404	12.399	11.994	12.470		20.009
6.291	11.067	12.062	11.099		23.004
4.550		15.776	10.251		22.236
8.033		22.469	11.369		16.309
7.748			4.644		16.407
			8.102		15.719

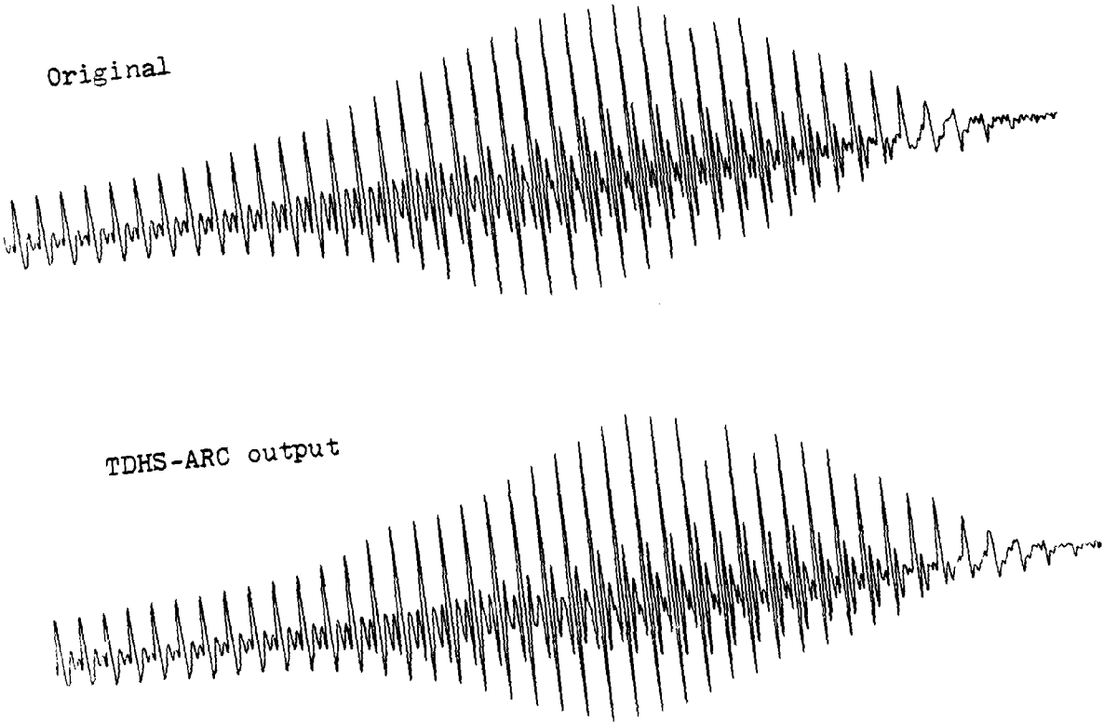


Fig. 4.4(a) Segment of female speech.

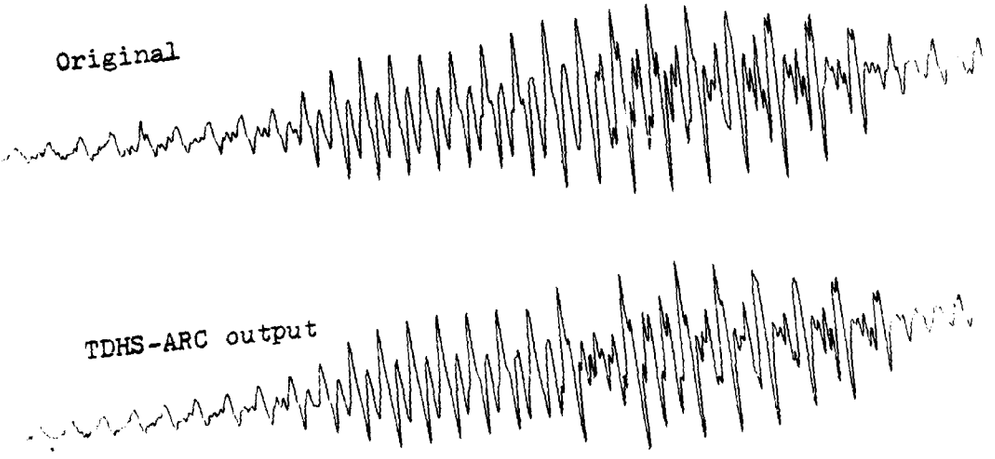


Fig. 4.4(b) Segment of male speech.

4.4 CODING

The ARC transmitter produces quantizer levels from 1 to 11. These source symbols need to be coded into channel alphabet for the transmission. The source coding must be such that it uses the bit rate optimally. The probabilities of the quantizer level occupancies are non-uniform. In such case, the bit rate is optimally utilized by assigning variable length code to the quantizer levels. The average code length for Huffman code, matches very well with the average entropy (Gallager, 1969). However, such simple variable length source code tended to cause the buffer to fill up rapidly during segments of voiced speech. When sophisticated variable length code, such as overfull code in PARC system [1980], is employed the effect of channel errors is severe. This is because a single bit error could lead to a string of inaccurate data. This becomes a serious disadvantage for a robust system. This leads to the development of a fixed length code.

With the compressed speech sampling rate of 3200 Hz and the bit rate of 9600 bits per second, the average number of bits per sample are 3. If simple fixed length 3 bit code words are used, only 8-level quantizer can be employed. Fewer quantizer levels cause more quantization noise, thus deteriorating the output speech quality. Therefore, 11-level quantizer was used. This requires 4-bit code word for fixed length coding. There are 16 possible code words out of which 11 are used to represent the quantizer levels and the rest is used to represent the run lengths as shown in Table 4.3. It can be noted there that a 4-bit code word can represent either one sample or two samples. Thus, either 4 bits or 2 bits per sample are transmitted as opposed to the average of 3 bits per

TABLE 4.3

Source code and a quantizer level statistics for a typical two second utterance.

Source Alphabet	Codewords		Frequency of Occupancy	Probability of Occupancy
1	0 0 0 0	0	516	0.1215
2	0 0 0 1	1	366	0.0862
3	0 1 0 0	4	351	0.0826
4	0 0 1 0	2	375	0.0883
5	0 1 0 1	5	265	0.0624
6	0 0 1 1	3	78	0.0184
7	0 1 1 0	6	79	0.0186
8	1 0 1 1	11	5	0.0012
9	0 1 1 1	7	26	0.0061
10	1 1 1 1	15	6	0.0000
11	1 1 1 0	14	7	0.0016
1, 1	1 0 0 0	8	920	0.2166
2, 1	1 0 1 0	10	402	0.0946
2, 2	1 1 0 1	13	248	0.0584
3, 1	1 0 0 1	9	432	0.1017
3, 3	1 1 0 0	12	178	0.0419

Probability of runlength occurring - 0.5132

Probability of no runlength - 0.4868

sample. If the probability of occurrence of run length is large enough, then the average number of bits per sample will be 3.

Let the probability of occurrence of run length be p. Hence (1-p) is the probability of no run length occurring.

Hence

$$\text{Average number of samples/bit} = \frac{p \cdot 2}{4} + (1-p)\frac{1}{4}$$

or

$$= \frac{1}{4}(1+p)$$

It is required that the average number of bits per sample should not exceed three or the average number of samples per bit should be more than one third.

Hence

$$\frac{1}{4}(1+p) > 1/3$$

or

$$p > 1/3$$

or the probability of runlength should be greater than 0.33

If the run lengths occur at least one third of the time, the average bit rate is maintained. The quantizer and the predictor parameters were designed such that lower level quantizer level occupancy is high enough to maintain the probability of run lengths around 0.33. Table 4.3 gives the statistics of the quantizer levels, run lengths and the code words. All the code words are 4 bits long and this prevents the propagation of

transmission error. However, samples can be added or deleted because of such error. Table 4.4 lists all the possible transitions for quantizer levels if the single bit error occurs. The third column shows the probability of the transition from run length to non-run length and vice-versa. To find out the average addition or deletion of samples at the receiver due to channel error, consider probabilities of the quantizer levels and that of the transitions given in Table 4.4.

Let ε be the bit error rate and p_i be the probability of ith quantizer level. Similarly, the run length probabilities are $p_{1,1}$, $p_{2,1}$, $p_{2,2}$, $p_{3,1}$ and $p_{3,3}$.

The samples added at the receiver per four bits transmitted =

$$\varepsilon \frac{1}{4}\{p_1+p_2+p_3+p_4+p_5+p_8+p_{10}+p_{11}\} + \frac{p_8}{4} + \frac{p_{11}}{4}$$

$$- \frac{1}{2}\{p_{1,1}+p_{2,1}+p_{2,2}+p_{3,1}+p_{3,3}\} + \frac{p_{1,1}}{4} - \frac{p_{2,1}}{4}$$

$$= \varepsilon \frac{(1-p)}{4} - \frac{p_6}{4} - \frac{p_7}{4} - \frac{p_9}{4} + \frac{p_8}{4} + \frac{p_{11}}{4} - \frac{p}{2} + \frac{p_{1,1}}{4} - \frac{p_{2,1}}{4}$$

or

$$= \varepsilon \frac{1-p}{4} - \frac{p}{2} + \frac{1}{4}\{p_8+p_{11}-p_{2,1}-p_6-p_7-p_9+p_{1,1}\}$$

where

p - probability of run length

(1-p) - probability of no run length

TABLE 4.4

Quantizer Level	Possible quantizer Level after error	Probability
1	[2,4,3,(1,1)]	1/4
2	1,6,5,(2,1)	1/4
3	5,7,1,(3,1)	1/4
4	6,1,7,(1,2)	1/4
5	3,9,2,(1,3)	1/4
6	4,2,9,8	0
7	9,3,4,11	0
8	(1,2),(2,1)10,6	1/2
9	7,5,6,10	0
10	11,(1,3),8,9	1/4
11	10,(3,1),(1,2),7	1/2
1,1	(2,1),(1,2),(3,1),7	1/4
2,1	8,(1,1),11,4	3/4
2,2	(3,1),10,(2,1),5	1/2
3,1	2,8,(1,3),(1,1)	1/2
3,3	3,11,(1,1),(1,3)	1/2

Consider the specific example, say the sentence 1, this number comes to 0.0006 samples for the BER of 1% or approximately 1.25 samples per second.

The bit string in this system represents only the quantizer levels. If this information is transmitted over noisy channel with BER as high as 0.1% without error protection, the output speech quality is slightly degraded. The higher BERs require error protection. (57,63) and (26, 31) single-error-correction Hamming codes were tried. The output speech quality was found to be much better for (26,31) single error correcting Hamming code. However, speech quality for no transmission error is sacrificed.

4.5 BUFFER CONTROL

It was noted in previous sections that the code employed in the system generates a variable number of bits per sample. This causes the buffer content to fluctuate considerably. Since the buffer is of fixed bit length, (1024 bits in this case) it is important to monitor the buffer behavior and control it if the buffer overflows or underflows. For example, in the unvoiced and silence segment of speech, more and more run lengths of quantizer levels are formed, thus generating only 2 bits per sample as against average rate of 3 bits per sample. If such situation continues for long time, the buffer may eventually run out of bits. The buffer overflow situation may occur for the voiced segment of speech where 4 bits per sample are generated as against 3 bits per sample are removed from the buffer.

The simulation of the bit buffer at the transmitter (or sample buffer at the receiver) is carried out by keeping count of the number of net bits (or samples) added to the buffer. The bit rate required for error protection is taken into account by adding (n-m) parity check bits to the transmitter buffer for m information bits. The sample buffer at the receiver is not affected by the error protection bits. The following equations describe the buffer simulations implementation

$$b(k) = b(k-1) + r(k) - \bar{r} \tag{4.1}$$

where

$b(k)$ = Current buffer content

k = Time instant

$r(k)$ = Instantaneous bit rate

= 4 bits/sample or 4 bits/2 samples

$\bar{r}$ = Average bit rate

$= \frac{m \times 3}{n}$ for (m, n) Hamming code

For no run length

$$b(k) = b(k-1) + 4 - \bar{r} \tag{4.2}$$

For run length

$$b(k) = b(k-1) + 4 - 2\bar{r} \tag{4.3}$$

Similarly, the receiver sample buffer simulation is expressed by

$$s(k) = s(k-1) + 0.25 - 0.333 \tag{4.4}$$

or

$$s(k) = s(k-1) + 0.50 - 0.333 \tag{4.5}$$

Equation (4.4) is for "no run length" case and Eq. (4.5) expresses "run length" situation. On an average, one sample per 3 bits or 1/3 sample per bit is added to the sample buffer. All the code words are four bit long. The number of samples added to the buffer is either one or two i.e. 0.25 or 0.5 samples per bit.

As mentioned earlier, the transmission buffer tends to underflow when large number of run lengths of quantizer levels is generated. The buffer underflow can be prevented by switching the code when small number of bits are left in it. Such switching can be accomplished by not employing a run length when the buffer contents drop below a certain threshold. In this case, the buffer content is incremented by one bit for every quantizer level. The buffer underflow control can be seen in Fig. 4.5. It shows the transmitter and receiver buffer behavior for an utterence spoken by a male speaker. When the bit count reaches one hundred, switching of code occurs and the bit count remains near or above the threshold.

The buffer overflow situation usually happens in the voiced segment of speech. The probability of run lengths is small when outer quantizer levels are frequently generated. In such case, there is a net addition of a bit for every sample, eventually causing overflow. The buffer overflow should be avoided since it could cause a loss of information. To prevent such situation, a method had to be found to sharply limit the bit generation rate occasionally which did not cause an unreasonable amount of distortion. One such method could be, changing the quantizer level thresholds gradually as the buffer fills beyond certain thresholds. In the simulations, three buffer thresholds and three scalars to change the

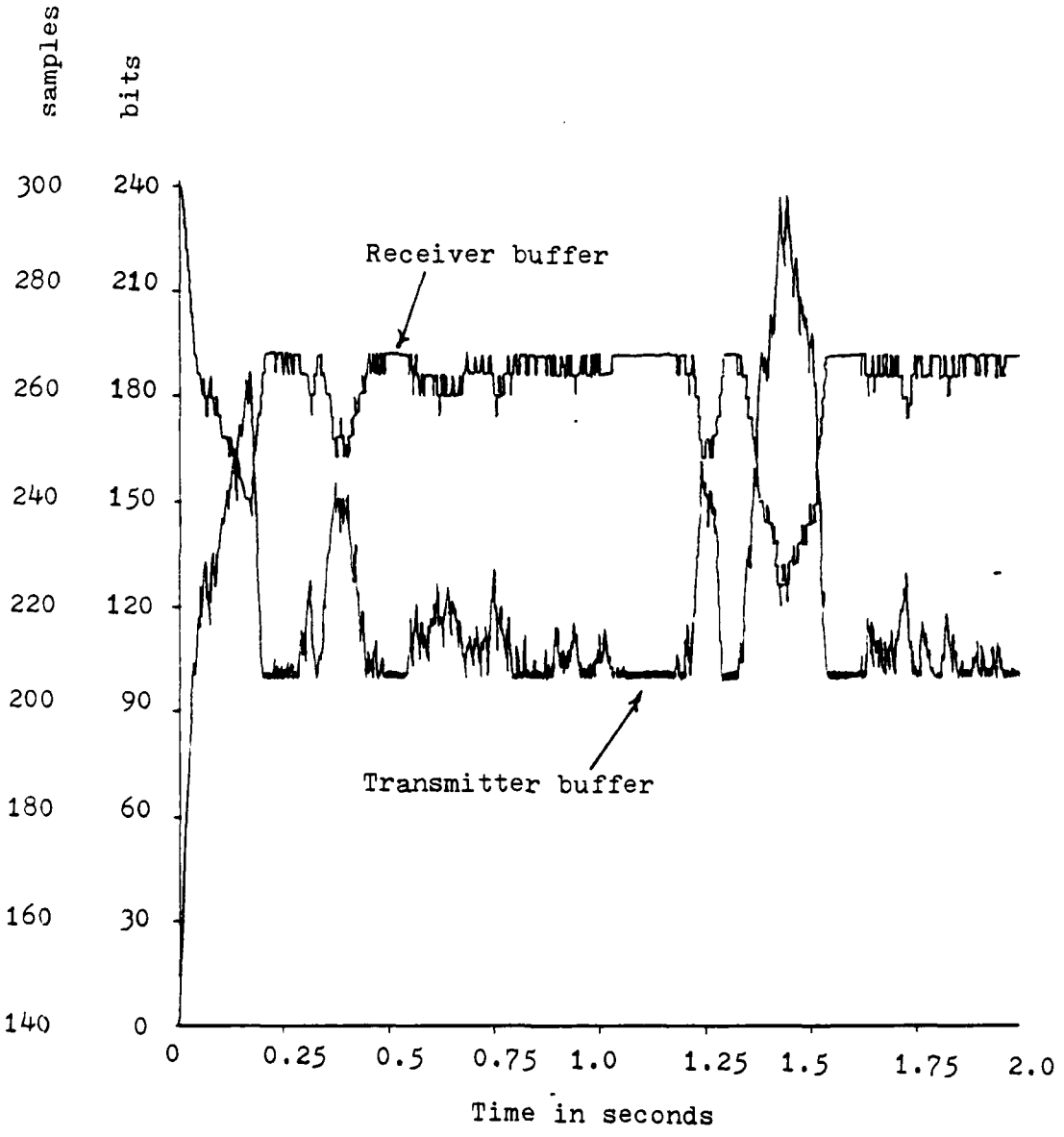


Fig. 4.5 Transmitter and receiver buffer behaviour.

quantizer level thresholds, were specified. The quantizer changes as the buffer crosses these thresholds, is shown in Fig. 4.6. This, of course, introduces more quantization noise and hence distortion. The performance of the ARC system with buffer control is outlined in Table 4.5. It can be seen that the distortion introduced is gradual. The transmitter buffer behavior with buffer control, can be seen in Fig. 4.7. The number of bits in the buffer is limited to 1024. The clipping of the buffer at this value indicates buffer overflow situation. With the buffer control strategy described above, more lower quantizer levels are generated thus increasing the run length probability. The Table 4.6 shows the statistics of the quantizer with and without the buffer control. The frequency of occupancies of lower levels is, indeed, increased and hence the probability of the run lengths. The advantage of this scheme is its simplicity since the receiver need not know the quantizer thresholds.

4.6 TRANSMISSION ERRORS

Many toll quality speech links maintain bit-error rates (BER) which are too small (less than 10^{-5}) to affect the quantizer and hence the coder performance. However, a BER of one tenth of a percent is not uncommon and for bad channels this rate could be as high as one percent. It is important to determine the extent of degradation and if possible how to minimize it.

The FORTRAN simulation program was written to study the effects of transmission errors. The bit manipulations required for the exact simulation of real situation is rather difficult to accomplish easily in

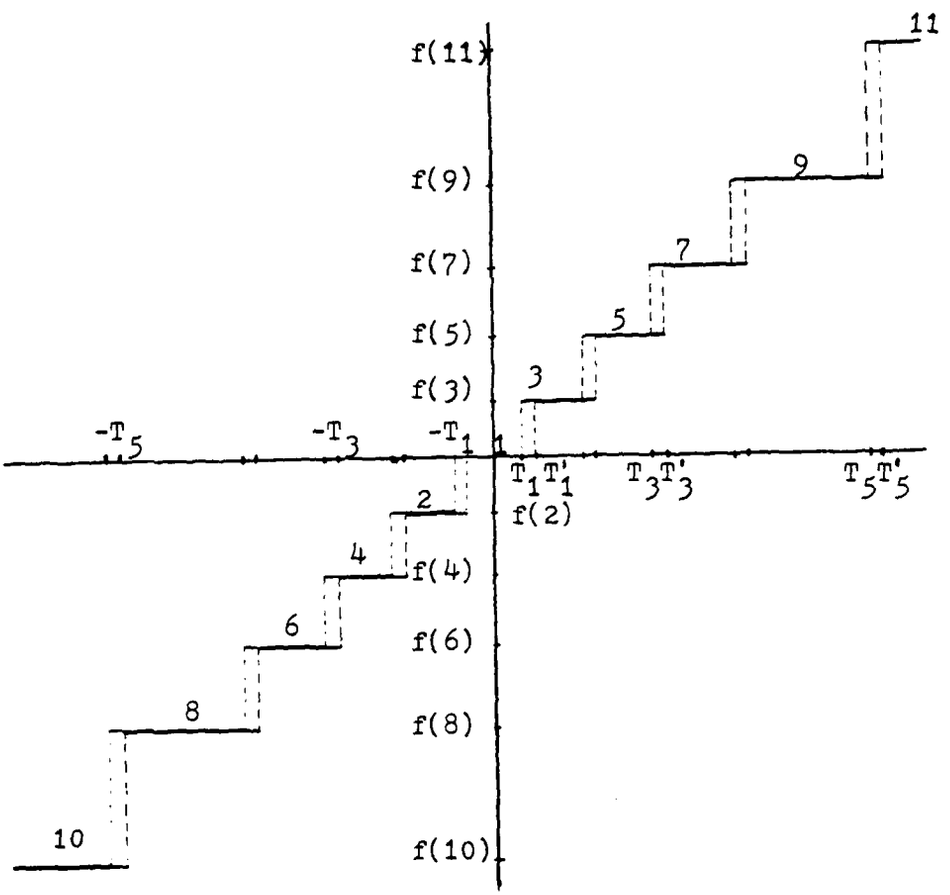


Fig. 4.6 The effect on the quantizer as the transmitter buffer gets full.

TABLE 4.5

The effect of buffer control on the ARC performance

Sample Number	ARC performance (dB)		Predictor Performance (dB)		Quantizer Performance (dB)	
	No Buffer Control	Buffer Control	No Buffer Control	With Buffer Control	No Buffer Control	With Buffer Control
4225- 4288	11.51	11.51	-2.15	-2.15	13.67	13.67
4289- 4352	11.Ø6	11.Ø6	-Ø.98	-Ø.98	12.Ø3	12.Ø3
4353- 4416	12.77	12.77	Ø.85	Ø.85	11.92	11.92
4417- 448Ø	11.76	11.76	1.83	1.83	9.93	9.93
4481- 4544	21.69	21.69	6.58	6.58	15.11	15.11
4545- 46Ø8	21.29	21.29	4.92	4.92	16.37	16.37
46Ø9- 4672	17.81	17.81	5.5Ø	5.5Ø	12.32	12.32
4673- 4736	18.65	16.44	6.31	6.Ø7	12.34	1Ø.37
4737- 48ØØ	15.Ø9	13.Ø6	2.96	2.5Ø	12.14	1Ø.55
48Ø1- 4864	1Ø.43	5.54	5.62	3.91	4.8Ø	1.63
4865- 4928	1.21	1.Ø4	1.21	1.Ø4	Ø.ØØ	Ø.ØØ
4929- 4992	Ø.ØØ	Ø.ØØ	Ø.ØØ	Ø.ØØ	Ø.ØØ	Ø.ØØ
4993 5Ø56	Ø.ØØ	Ø.ØØ	Ø.ØØ	Ø.ØØ	Ø.ØØ	Ø.ØØ
5Ø57- 512Ø	17.2Ø	17.24	2.34	2.34	14.86	14.89
5121- 5184	1Ø.28	1Ø.28	Ø.86	Ø.86	9.42	9.42
5185- 5248	18.85	18.86	4.21	4.21	14.64	14.64
5249- 5312	16.85	16.85	6.24	6.24	1Ø.61	1Ø.61
5313- 5376	17.84	17.84	6.47	6.47	11.37	11.37

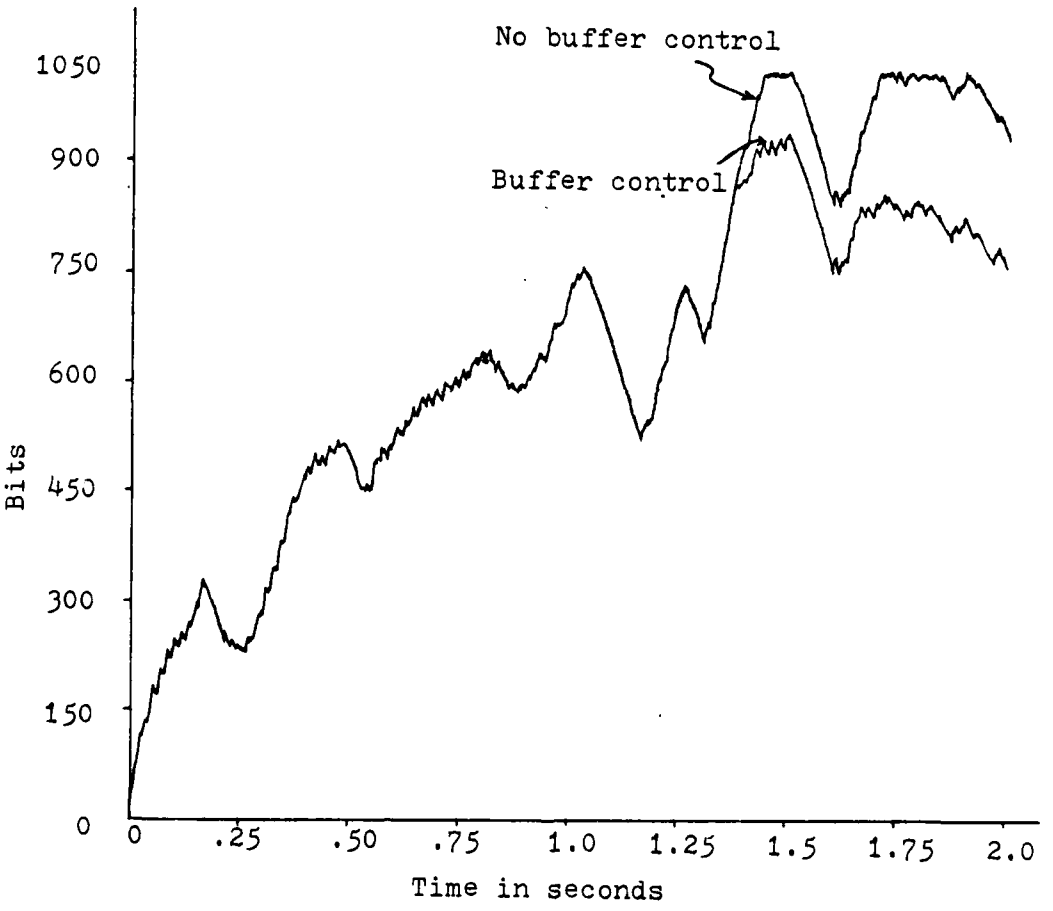


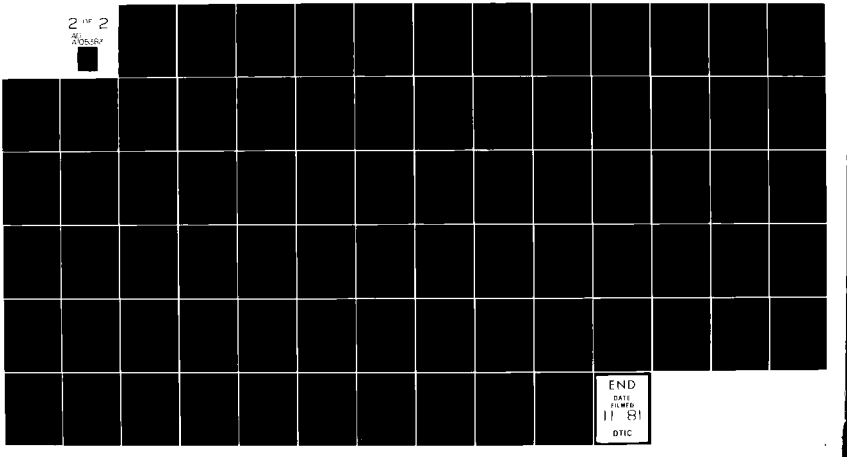
Fig. 4.7 Transmitter buffer behaviour with buffer control

TABLE 4.6

The statistics of quantizer levels with and without buffer control

Source Alphabet	No buffer control		With buffer control	
	Frequency	Probability	Frequency	Probability
1	516	Ø.1215	525	Ø.1242
2	366	Ø.Ø862	34Ø	Ø.Ø8Ø5
3	351	Ø.Ø826	332	Ø.Ø786
4	375	Ø.Ø883	373	Ø.Ø883
5	265	Ø.Ø624	265	Ø.Ø627
6	78	Ø.Ø184	71	Ø.Ø168
7	79	Ø.Ø186	8Ø	Ø.Ø189
8	5	Ø.ØØ12	6	Ø.ØØ14
9	26	Ø.ØØ61	25	Ø.ØØ59
1Ø	Ø	Ø.ØØØØ	Ø	Ø.ØØØØ
11	7	Ø.ØØ16	7	Ø.ØØ17
1 1	92Ø	Ø.2166	997	Ø.2359
2 1	4Ø2	Ø.Ø946	394	Ø.Ø932
2 2	248	Ø.Ø584	234	Ø.Ø554
3 1	432	Ø.1Ø17	413	Ø.Ø977
3 3	178	Ø.Ø419	164	Ø.Ø388

AD-A105 383 NOTRE DAME UNIV IN DEPT OF ELECTRICAL ENGINEERING F/G 9/2
NEW APPROACH TO SPEECH DIGITIZATION COMBINING TIME-DOMAIN HARMO--ETC(U)
AUG 81 J L MELS, A K PANDE DCA100-80-C-0050
UNCLASSIFIED NL



FORTRAN. However, the following steps in simulation procedure do represent the real time situation very closely.

1. The encoder program reads a block of 256 quantizer levels, converts them into code words and writes them into a file. This file length is always shorter than the quantizer level file because of run lengths.
2. The channel simulation program reads the code words and asks the BER during run time. Depending on the choice of n in (m, n) Hamming code, the block of code words is chosen. For example, (57,63) single error-correcting Hamming code, the block of 16 code words is considered. The errors are added in the bit stream according to the bit error rate specified. The single error is corrected, the double or even number of errors are passed uncorrected and for three or more odd numbers of errors additional error is introduced. The single error corrected code words are written in a separate file. The decoder program reads this file and produces a quantizer level file which forms the input to the receiver program.

The simulation described was carried out with the parameters designed to minimize the effect of transmission errors. [Melsa, et al., 1980]. Because of addition and deletion of samples due to the channel errors the system performance cannot be reliably evaluated using objective measure criteria. Informal listening test was used to measure the performance. The output speech is distorted due to two reasons. First, due to the error in quantizer levels and second, due to wrong pitch

extraction. As mentioned in an earlier chapter, the pitch is extracted from the reconstructed compressed speech, y(k). If y(k) is changed significantly due to the channel errors, the wrong pitch might be extracted. Table 4.8 shows the pitch extracted at the transmitter and at the receiver with and without channel errors. It also lists the pitch periods extracted after the error correction. The effect of 1% channel errors on the pitch extraction is not significant. Besides, the distortion caused due to the use of a wrong pitch period is masked by that due to the channel error. After error correction at the receiver, pitch is extracted generally correctly and thus, pitch extraction at the receiver does not cause any severe degradation in the presence of channel noise.

In the previous section, the effects of [26,31] and [57,63] single error correcting Hamming code on the buffer behavior were discussed. The allocation of significant bit rate for error protection penalizes the error free performance because of the repeated use of the buffer overflow control strategy. However, the system performance in the presence of channel noise is greatly improved. The extent of this improvement can be determined by finding BER after using single error correcting Hamming code.

Let [m,n] be the single error correcting Hamming code and ε be the BER.

$$P\{\text{no error per frame}\} = (1-\varepsilon)^n$$

$$P\{\text{at least one error per frame}\} = 1 - (1-\varepsilon)^n$$

$$P\{\text{single error per frame}\} = n(\varepsilon)(1-\varepsilon)^{n-1}$$

$$P\{\text{double error per frame}\} = \frac{n(n-1)}{1x2} \ (\varepsilon)^2(1-\varepsilon)^{n-2}$$

TABLE 4.8

Effect of channel errors on pitch extraction

Pitch at the transmitter	Pitch at the receiver No channel error	Pitch at the receiver 1% BER	Pitch at the receiver 1% BER with single error correction	Pitch at the receiver 1% BER with double error correction
5333 - 5632 29	2664 - 2863 29	2649 - 2848 9#	267# - 2869 29	2641 - 284# 89
5391 - 569# 3#	2693 - 2892 3#	2739 - 2938 3#	2699 - 2898 3#	273# - 2929 3#
5451 - 565# 3#	2723 - 2922 3#	2769 - 2968 3#	2729 - 2928 3#	276# - 2959 3#
5511 - 571# 3#	2753 - 2952 3#	2799 - 2998 31	2759 - 2958 3#	279# - 2989 3#
5571 - 577# 3#	2783 - 2982 3#	283# - 3#29 29	2789 - 2988 3#	282# - 3#19 3#
5631 - 583# 3#	2813 - 3#12 3#	286# - 3#49 3#	2819 - 3#18 3#	285# - 3#49 3#
5691 - 589# 3#	2843 - 3#42 3#	288# - 3#79 3#	2849 - 3#48 3#	288# - 3#79 3#
5751 - 595# 3#	2873 - 3#72 3#	291# - 31#9 3#	2879 - 3#78 3#	291# - 31#9 3#
5811 - 6#1# 3#	29#3 - 31#2 3#	294# - 3139 3#	29#9 - 31#8 3#	294# - 3139 3#
5671 - 6#7# 3#	2933 - 3132 3#	297# - 3169 3#	2939 - 3138 3#	297# - 3169 29
5931 - 613# 3#	2963 - 3162 3#	3### - 3199 3#	2969 - 3168 3#	2999 - 3198 29
5991 - 619# 3#	2993 - 3192 3#	3#3# - 3229 3#	2999 - 3198 3#	3#28 - 3227 29
6#51 - 625# 3#	3#23 - 3222 3#	3#6# - 3259 3#	3#29 - 3228 3#	3#57 - 3256 3#
6111 - 631# 3#	3#53 - 3252 3#	3#9# - 3289 3#	3#59 - 3258 3#	3#87 - 3286 3#
6171 - 637#. 3#	3#83 - 32#2 3#	312# - 3319 3#	3#89 - 3288 3#	3117 - 3316 3#
6231 - 643# 3#	3113 - 3312 3#	315# - 3349 3#	3119 - 3318 3#	3147 - 3346 3#
6291 - 649# 3#	3143 - 3342 3#	318# - 3379 29	3149 - 3348 3#	3177 - 3376 3#
6351 - 655# 3#	3173 - 3372 3#	32#9 - 34#8 89	3179 - 3378 3#	32#7 - 34#6 3#
6411 - 661# 3#	32#3 - 34#2 3#	3298 - 3497 29	32#9 - 34#8 89	3237 - 3436 29
6471 - 667# 3#	3233 - 3432 3#		3298 - 3497 3#	

If a single error in the frames is corrected, the residual error becomes $[1-(1-\varepsilon)^n-n\varepsilon(1-\varepsilon)^{n-1}]$. This error could be equivalent to the BER, δ, without any error correction. This is expressed in the following equations.

$$1 - (1-\delta)^n = 1 - (1-\varepsilon)^{n-1} - n\,\varepsilon\,(1-\varepsilon)^{n-1} \qquad (4.5)$$

let $n = 31$ and $\varepsilon = 0.01$ or BER = 1%

$$[1 - (1-\delta)^{31}] = [1 - (0.99)^{31}] - 31(.01)\,(.99)^{30}$$

$$= 0.2676 - .2293 = .03839$$

$$(1-\delta)^{31} = 0.9616$$

$$31\ \ln(1-\delta) = \ln 0.9616$$

$$\ln(1-\delta) = -.00126$$

$$1-\delta = 0.9987 \quad \text{or} \quad \delta = 0.0012$$

i.e. 1% BER with single error correction using [26,31] Hamming code is equivalent to 0.12% BER without any correction. The similar calculations and the simulation results, for different BERs and Hamming codes, are outlined in Table 4.9. It can be seen that [26,31] single error correcting Hamming code is quite effective against transmission errors. While studying the Table 4.9 it should be kept in mind that the frame size used in the simulation studies, was 64 and 32 bits instead of the actual 63 and 31 bits respectively. Also, it should be noted in the simulation studies that additional errors were introduced if 3 or more odd number of errors were detected in a frame.

It was mentioned earlier that transmission errors cause addition or deletion of samples. This happens because the four-bit code word represent either one or two samples. If the channel error changes the code

TABLE 4.9

Sentence 1

Male speaker, 2 second utterance, sampling frequency = 3200 Hz

BER	Hamming Code	Theoretical BER after single error correction	Simulated BER after single error correction	Theoretical BER after double error correction	Simulated BER after double error correction
0.1%	[57,63]	0.0029%	0.074%	0.097%	0.092%
	[26,31]	0.0014%	0.074%	0.0985%	0.092%
1.0%	[57,63]	0.26%	0.502%	0.714%	0.553%
	[26,31]	0.137%	0.307%	0.85%	0.713%
5.0%	[57,63]	2.77%	4.8%	2.15%	3.6%
	[26,31]	1.98%	4.16%	2.42%	3.31%

word to represent two samples instead of one, as it should be, there is an addition of samples at the receiver. The deletion of sample occurs if the reverse situation happens. In the earlier discussion of buffer control, it was pointed out that the receiver buffer tends to overflow during silence and underflows during voiced region. A desirable effect of a channel error would be the addition of samples. This might cause deletion of silence or prevent underflowing of the buffer during voiced segment. For the two second utterence, the effect of channel errors on the receiver buffer behavior is not so pronounced. Therefore, the BER of 5% was considered. The receiver buffer plot for five percent bit error rate is shown in Fig. 4.8. It can be clearly seen that the net effect of channel errors is to add samples to the buffer for the code word assignment in Table 4.3.

4.7 BACKGROUND NOISE

The system performance was evaluated for the male and female utterances, spoken in quiet rooms without any background noise. So, the input to the coder is undistorted input. However, this is an unlikely situation since there will certainly be background interference such as other speakers, typewriter noises and the like, whenever a speaker is using the "digital telephone". It was thought that the periodic background noise would be the worst kind of noise for TDHS-ARC system since the periodicity of the input signal is used for harmonic scaling operations.

The study of background noise is very simple after the algorithm has been implemented in real time. It is just a matter of talking into a handset with noise in the background. The output could be heard through headphones. However, in the FORTRAN simulation, the task is not so

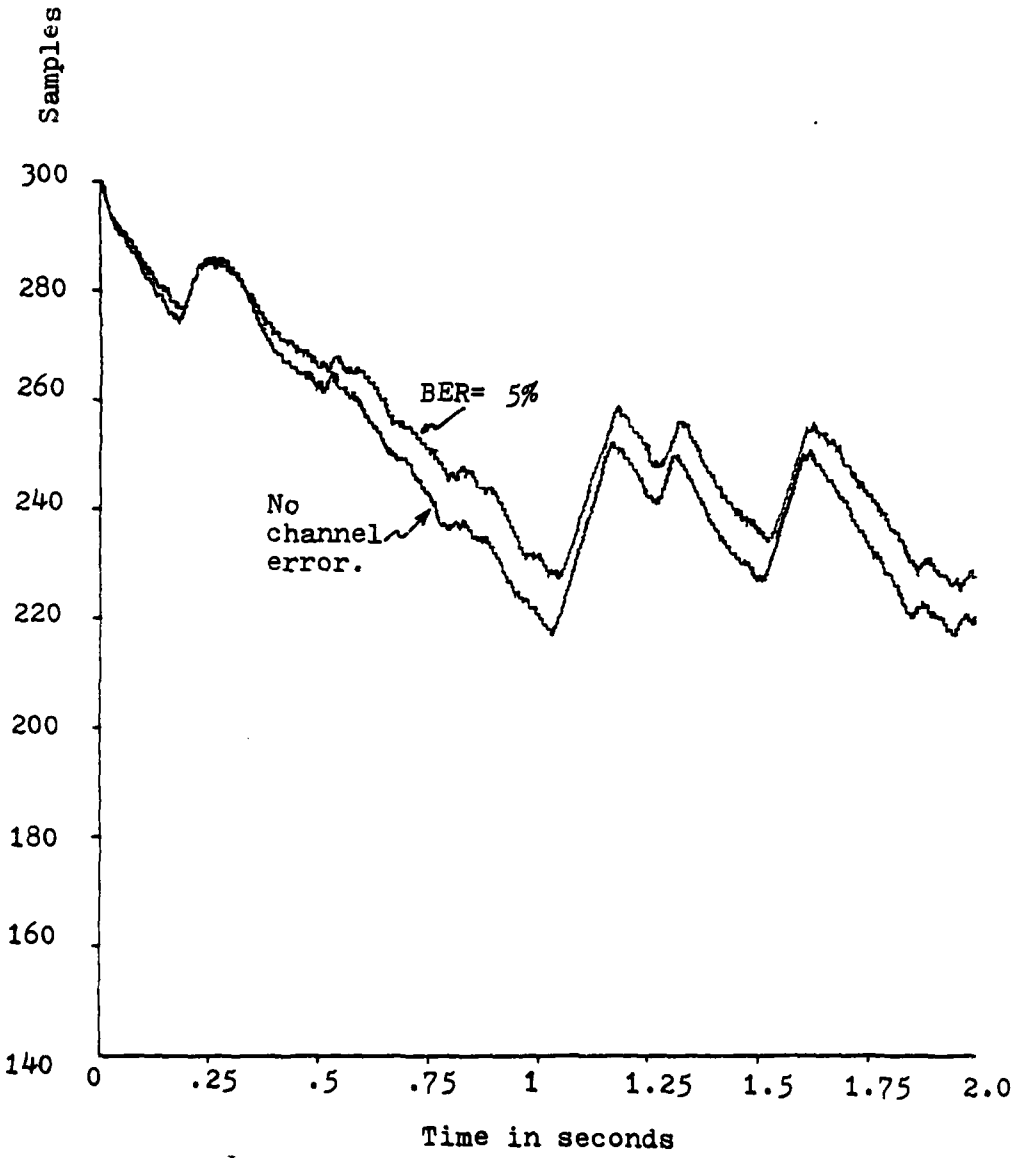


Fig. 4.8 The effect of channel error on a receiver buffer.

straightforward. There is a need for a digital speech file with background noise. Multispeaker files were created by adding two digital speech files with appropriate weight.

In the simulation, the multispeaker file was generated by adding female speech to male speech as in Eq. 4.6

$$s_{composite} = s_{11} + k\, s_1 \tag{4.6}$$

where k takes values from 0 to 1 thus having varying degree of background noise. It was noticed that pitch extraction loop picks the pitch for the dominant speaker (see Table 4.10) at each short-time interval. Due to the masking properties of the ear, the harmonic distortion in the non-dominant speech is not heard. The ARC and the total system performance for the composite speaker is shown in Table 4.11. As long as the pitch tracking algorithm does not break down, this system should perform very well for background noise.

4.8 SUMMARY

The system presented in this chapter produces a high quality speech at the transmission rate of 9.6 kb/s. The speech quality, however, depends on the bit rate used for error protection. For example, the use of [26,31] single error correcting Hamming code results in an excellent system performance for the bit error rate of 1%. However, noise free performance is degraded because of coarse quantization and/or buffer control operations. The quantizer level runlength is effectively used to employ fix codeword size. The switching of code and changing the quantizer thresholds was found to be the effective strategy to control buffer underflow and overflow respectively. The high bit error rates do affect

TABLE 4.10

Pitch extraction for composite speaker

$S_{11} + 0.5\ S_1$		S_{11}		S_1	
Sample #	Pitch	Sample #	Pitch	Sample #	Pitch
3953 - 4092	57	3939 - 4078	88	3939 - 4078	57
4067 - 4206	58	4115 - 4254	29	4053 - 4192	58
4183 - 4322	58	4173 - 4312	58	4169 - 4308	58
4299 - 4438	59	4289 - 4428	29	4285 - 4424	59
4417 - 4556	59	4347 - 4486	29	4403 - 4542	60
4535 - 4674	60	4405 - 4544	29	4523 - 4662	60
4655 - 4794	62	4463 - 4602	87	4643 - 4782	62
4779 - 4918	65	4637 - 4776	29	4767 - 4906	65
4909 - 5048	67	4695 - 4834	29	4897 - 5036	66
5043 - 5182	64	4753 - 4892	29	5029 - 5168	65
5171 - 5310	100	4811 - 4950	28	5159 - 5298	64
5371 - 5510	29	4867 - 5006	98	5287 - 5426	65
5429 - 5568	30	5063 - 5202	65	5417 - 5556	20
5489 - 5628	61	5193 - 5332	58	5457 - 5596	66
5611 - 5750	30	5309 - 5448	26	5589 - 5728	66
5671 - 5810	30	5361 - 5500	29	5721 - 5860	52
5731 - 5870	30	5419 - 5558	30	5825 - 5964	67
5791 - 5930	60	5479 - 5618	30	5959 - 6098	34
5911 - 6050	30	5539 - 5678	30	6027 - 6166	53
5971 - 6110	30	5599 - 5738	30	6133 - 6272	35
6031 - 6170	30	5659 - 5798	30	6203 - 6342	72
6091 - 6230	30	5719 - 5858	30	6347 - 6486	29
6151 - 6290	30	5779 - 5918	60	6405 - 6544	53
6211 - 6350	30	5899 - 6038	30	6511 - 6650	68
6271 - 6410	30	5959 - 6098	30	6647 - 6786	31
6331 - 6470	30	6019 - 6158	30	6709 - 6848	48
6391 - 6530	30	6079 - 6218	30		
6451 - 6590	30	6139 - 6278	30		
6511 - 6650	30	6199 - 6338	30		
6571 - 6710	30	6259 - 6398	30		
6631 - 6770	30	6319 - 6458	30		
6691 - 6830	30	6379 - 6518	30		
6751 - 6890	30	6439 - 6578	30		
		6499 - 6638	30		
		6559 - 6698	30		
		6619 - 6758	30		
		6679 - 6818	30		
		6739 - 6878	30		

TABLE 4.11

Performance of ARC and TDHS-ARC system for multispeaker files

Sentence #	ARC			TDHS - ARC
	SNR (dB)	SPER (dB)	SNRQ(dB)	SEGSNR (dB) in Freq. domain
S_{11}	15.29	5.68	9.61	13.02
$S_{11}+.25S_1$	16.82	6.09	10.73	11.91
$S_{11}+.5S_1$	17.08	6.11	10.97	11.51
S_1	13.98	4.60	9.38	8.32

the pitch extraction at the receiver. However, the distortion caused by improper pitch period was found to be masked by the distortion due to transmission error effects. The system also behaves very well for simultaneous speakers.

CHAPTER 5

THE 16 KB/S SYSTEM

5.1 INTRODUCTION

In the earlier chapters, the TDHS-ARC system for the bit rate of 9.6 kb/s was presented. The same system design can be extended to a bit rate of 16 kb/s with a few modifications. The speech quality and the robustness of the 16 kb/s system are improved with the availability of more bits per sample for coding. The speech quality improvement is achieved by reducing the quantiziation noise, which can be done by increasing the number of quanitzer levels. The amount of error protection available determines the robustness of the system. These two basic issues concerning 16 kb/s system are discussed in detail in this chapter. The other aspects of the system design such as the system configuration, the buffer control strategy and the type of source code, remain the same.

Section 5.2 describes the source and the channel code used in this system. The choice of parameters in the ARC design and the buffer behaviour is discussed in Section 5.3. The effect of transmission errors is also discussed in this section. The results are summarized in Section 5.4.

5.2 SOURCE AND CHANNEL CODING

With the bit rate of 16 kb/s and the sampling frequency of 3.2 kHz of the compressed speech, an average of 5 bits per sample are available for coding. A 31-level quantizer and 5-bit fixed length codewords were used. The system performance was found to be excellent and no distortions could be heard in the informal listening tests. However, the above scheme

of coding is possible only if no error protection is desired. For the noisy channel, a 21-level quantizer with the 5-bit fixed wordsize variable input code was designed. Out of the possible 32 codewords, 21 codewords are used to represent the quantizer levels and the remaining are used to represent the runlengths. The lower quantizer levels (for example: 1,2, 3,4) form the runlengths. The code designed to represent the quantizer levels and the runlengths is shown in Table 5.1. The Hamming distance between the codewords representing adjacent quantizer levels is kept to be minimum possible. With the choice of codewords as shown in Table 5.1, the buffer empties by 2.5 bits per sample, everytime the runlength occurs while there is no net addition to the buffer otherwise. If the error protection is used, the check bits are added to the buffer every certain number of the information bits. This may cause a buffer overflow depending on the amount of error protection used. Such situations may be avoided by adjusting the probability of occurrence of runlengths.

If the probability of occurrence of runlength is p, then (1-p) becomes the probability of no runlength occurring. Let [m,n] be the Hamming code used, where m are the information bits and (n-m) are the check bits. The bit rate available to code the quantizer levels becomes $16 \cdot m/n$ kb/s and the average number of bits per sample is $5 \cdot m/n$. With the above probabilities,

$$\text{the average number of samples/bits} = \frac{p \cdot 2}{5} + \frac{(1-p)}{5}$$

or

$$= \frac{1}{5}(1+p)$$

It is required that the average bits per sample be equal or less than $5m/n$ or the average number of samples per bit be equal or more than $n/5m$.

TABLE 5.1

The Codeword Assignment

Source Alphabet	Codewords
1	00000
2	10000
3	00001
4	10001
5	00011
6	10011
7	00010
8	10010
9	00110
10	10110
11	00111
12	10111
13	00101
14	10101
15	00100
16	10100
17	01100
18	11100
19	01101
20	11101
21	01111
1,1	01000
1,2	11000
1,3	01001
2,1	11001
2,2	11011
2,3	11010
3,1	01011
3,2	01010
3,3	01110
4,4	11110
5,5	11111

Hence

$$\frac{1}{5}(1+p) > \frac{n}{5m} \qquad (5.1)$$

or

$$p > \frac{n-m}{m} \qquad (5.2)$$

For the Hamming code of [11,15] and [26,31] this probability should be equal to or greater than 0.36 and 0.19 respectively. That means, 36% or 19% of the time runlengths should occur to maintain the average number of bits per sample. The quantizer and the parameters in the ARC system were designed to satisfy Eq. (5.2). Table 5.2 shows the statistics and the parameters used for two different Hamming codes. The block-to-block SNR for ARC is plotted for both the parameter sets in Fig. 5.1. It can be observed from this figure that the heavy error protection leads to more quantization noise since less bit rate is available for the quantizer level coding.

It was discussed for the 9.6 kb/s system that the compromise between the system robustness and the speech quality must be made. It is desired in this project to have a good system performance for the bit-error-rate (BER) of 1%. If [11,15] single error correcting Hamming code is used and the computations as in Section 4.6 are carried out, 1% BER with a single error correlation becomes equivalent to 0.06% BER, with no error correction. For [26,31] Hamming code 1% BER becomes equivalent to 0.12% BER with no error correction. In the simulation studies it was found that the effect of BER of 0.06% was hardly noticeable.

5.3 BUFFER BEHAVIOUR

There are two buffers in the system. The transmitter buffer which is a bit buffer and the receiver sample buffer. The simulation of these buffers

TABLE 5.2

The parameters and the quantizer level statistics for [11,15] and [26,31] single error correcting Hamming codes for a two second male utterance.

For [11,15] Hamming Code						For [26,31] Hamming Code					
AINV = .7, ALP = .93, ALAD = .95, G = .018, N = 4 RMSMIN = 70, SMIN = .28 Number of quantizer levels = 21 Expansion factors: .5, .6, .7, .8, 1.0, 1.0 1.2, 1.4, 2, 4, 6 Output levels: 0, 0.6, 0.8, 1.1, 1.5, 2, 3, 5, 8, 12, 18						AINV = .7, ALP = .93, ALAD = .95, G = .018, N = 4 RMSMIN = 30, SMIN = .28 Number of quantizer levels = 21 Expansion factors: .5, .6, .7, .8, 1, 1, 1.2, 1.4, 2, 4, 6 Output levels: 0, .4, .6, .8, 1.0, 1.4, 1.8, 2.5, 5, 8, 12					
Source Alaphabet	Proba-bility	Fre-quency	Source Alaphabet	Proba-bility	Fre-quency	Source Alaphabet	Proba-bility	Fre-quency	Source Alaphabet	Proba-bility	Fre-quency
1	.0626	296	17	.0051	24	1	.0611	326	17	.0127	68
2	.0389	184	18	.0002	1	2	.0504	269	18	.0002	1
3	.0395	187	19	.0011	5	3	.0414	221	19	.0034	18
4	.0507	240	20	.0000	0	4	.0523	279	20	.0000	0
5	.0567	268	21	.0000	0	5	.0515	275	21	.0009	5
6	.0615	291	1,1	.1584	749	6	.0484	258	1,1	.1799	426
7	.0647	306	1,2	.0336	159	7	.0517	276	1,2	.0152	81
8	.0543	257	1,3	.0277	131	8	.0583	311	1,3	.0154	82
9	.0524	248	2,1	.0296	140	9	.0684	365	2,1	.0197	105
10	.0444	210	2,2	.0152	72	10	.0544	290	2,2	.0114	61
11	.0372	176	2,3	.0142	67	11	.0549	293	2,3	.0101	54
12	.0319	151	3,1	.0346	161	12	.0534	285	3,1	.0169	90
13	.0262	124	3,2	.0127	60	13	.0435	232	3,2	.0069	37
14	.0104	49	3,3	.0178	84	14	.0452	241	3,3	.0103	55
15	.0097	46	4,4	.0044	21	15	.0422	225	4,4	.0067	36
16	.0000	0	5,5	.0049	23	16	.0064	34	5,5	.0067	36

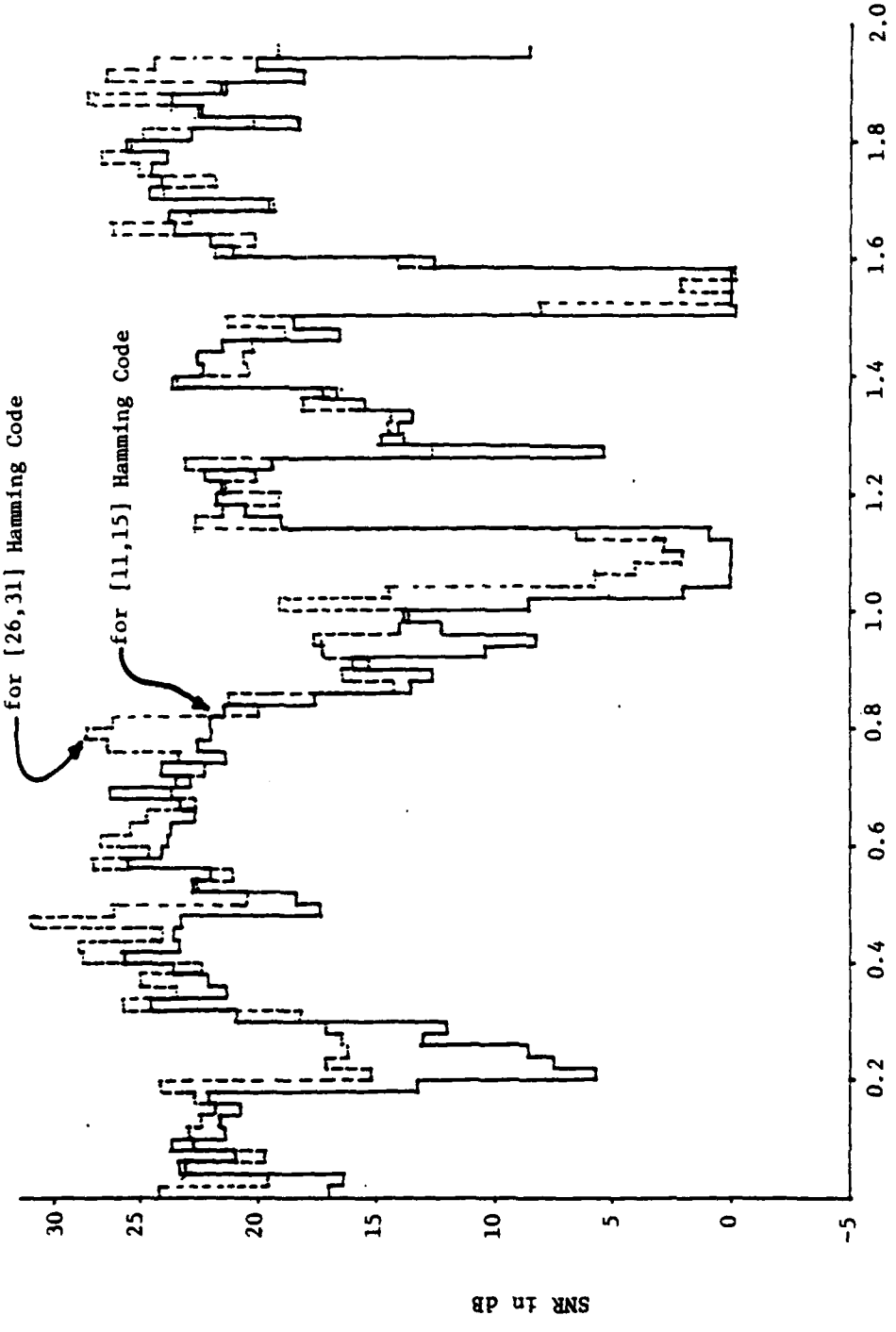


Fig. 5.1 The ARC performance for a male speaker using [11,15] and [26,31] Hamming Code.

is carried out the similar way as in 9.6-kb system. For no runlength case, 5 bits are added to the buffer while the same number of bits are taken out of the buffer. Thus, there is no net addition to the buffer. For the runlength (of two samples) case, 5 bits are added as against 10 bits are taken out of the buffer. In addition to this (n-m) checkbits are added to the buffer for every m information bits.

The decoder at the receiver decodes the frame of bits, makes the corrections if any, and puts the samples in the receiver buffer. For 16-kb system, the average number of samples per bit taken out of the receiver buffer is 1/5. If [11,15] Hamming code was used, for every frame of 15 bits 3 samples are taken out and the number of samples varying from 2.2 to 4.4 are added to the buffer. To work with the integer number of samples, the receiver buffer calculations are done every 75 bits or every 4.67 msec for [11,15] code and every 155 bits (≈9.67 msec) for [26,31] code. The transmitter and the receiver buffer plots are shown in Fig. 5.2(a) and (b) for both the codes. For the same set of parameters the transmitter buffer fills faster for [11,15] code than for [26,31] code thus requiring buffer control more frequently.

5.4 SUMMARY

A speech coder was developed for transmission of speech at the bit rate of 16 kb/s using time domain harmonic scaling and adaptive residual coding. The system configuration is the same as 9.6 kb/s system. It was decided to avoid pitch transmission since it retains the simplicity of the system. Besides, the distortion introduced by extracting the pitch information at the receiver, is masked by the quantization noise. The bit rate, which would have been wasted in transmitting and error protecting

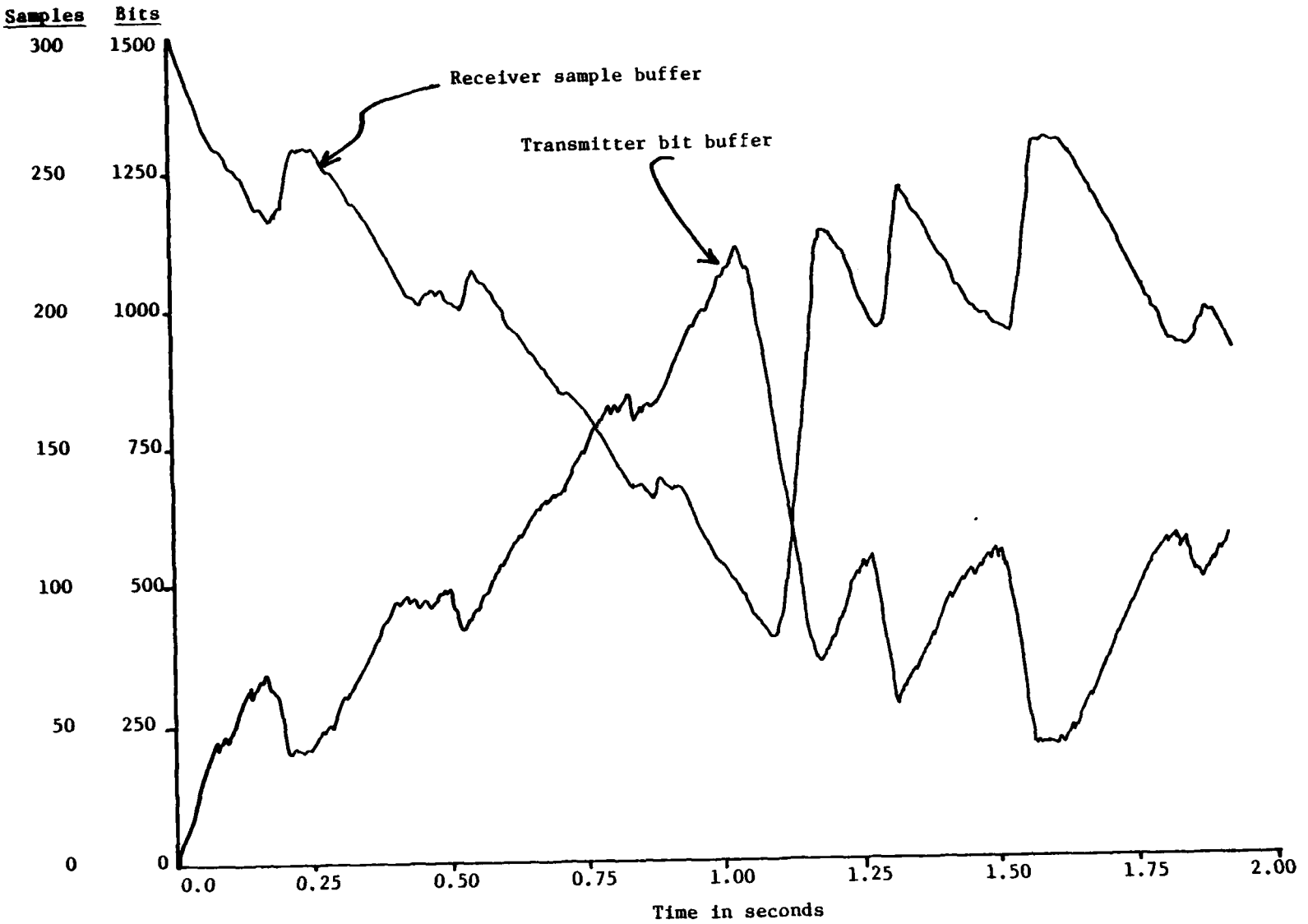


Fig. 5.2(a) Transmitter and receiver buffer plots for a male speaker using [26,31] Hamming Code

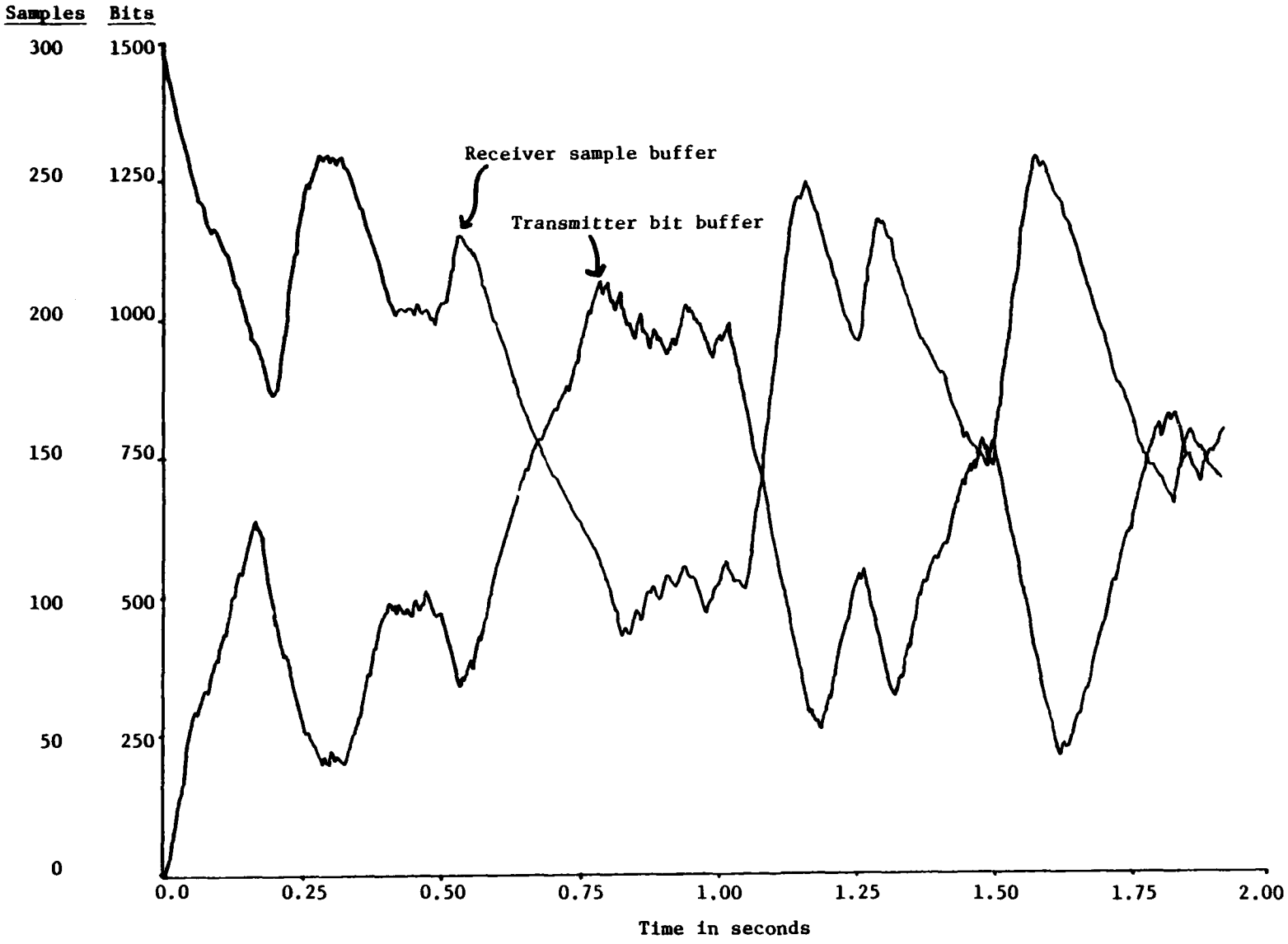


Fig. 5.2(b) Transmitter and receiver buffer plots for a male speaker using [11,15] Hamming Code

the pitch information, could be employed to make the system more robust to channel noise. Thus, the use of [11,15] Hamming code was possible. A segmental SNR of more than 20 dB was achieved using 21-level quantizer. The variable input code of 5-bit fixed wordsize was found to be useful since the transmission error effects do not get magnified.

An attempt is made to keep the structure of the 16 kb/s system as simple as 9.6 kb/s system. No side information is transmitted, hence there is no blocking of data or block synchronization problem. The frequency compression and expansion operations by a factor of two are carried out using a triangular window. The hardware implementation of the system should be the same as a 9.6 kb/s system except for the quantizer design and the codeword size.

CHAPTER 6

SUMMARY

A speech coder was developed using a new approach of combining frequency scaling in time domain and adaptive residual coding. The computer simulations of the system were kept very close to real situation. The speech quality was found to be excellent. The system's overall performance could not be measured satisfactorily using existing objective performance measure criterion. In such cases, the output speech quality was evaluated by using informal listening tests. However, there are various objective measure criteria as listed in Chapter 3 to evaluate the performance of DPCM coders. Various block-to-block SNR plots indicate that the adaptive quantizer and the predictor perform very well. Such excellent performance of the ARC system is possible because of smooth varying frequency compressed input signal.

This system is designed for the bit error rate of 1%. However, it can easily withstand bit error rates higher than 5%. In such a severe channel condition, the output speech is considerably distorted but the algorithm does not diverge. In the simulation studies, it was noticed that a compromise must be reached between robustness and transmission error free coder performance. With the telephone modem assuring bit error rates less than 10^{-3}, output speech quality is very close to toll quality. The effect of channel error is reduced, particularly in this system, because of averaging operations performed at the receiver to get expanded speech. The transmission error effects do not get magnified because of the fixed word size of the codeword. In many coders for bit rate of 9.6 kb/s, entropy coding is used. Variable length codewords, such as Huffman code,

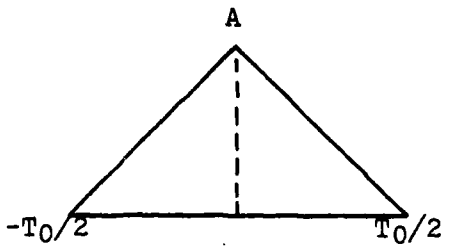
though optimally utilize the available bit rate, are very inefficient if the channel error occurs. One bit error could cause a string of wrong code words to be decoded.

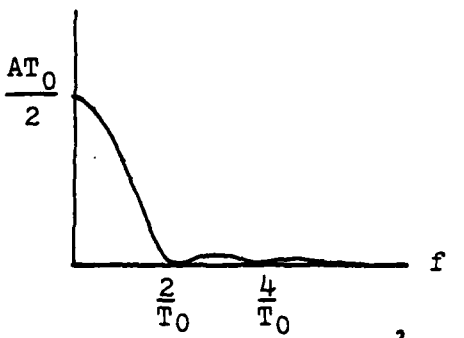
Another coder robustness indicator is its performance in background noise. In real situation, the talker is often talking in the presence of typewriter noise or background conversation. This coder performs very well for the multispeaker case. The waveform coders generally have this advantage over frequency domain speakers. It was observed in our simulation studies that various tones pass through this system with slight or no distortion. This excellent performance for tones is due to the fact that harmonic scaling operations do not introduce any distortion for perfectly periodic signals.

APPENDIX A

FOURIER TRANSFORM OF WINDOW FUNCTIONS

Triangular window

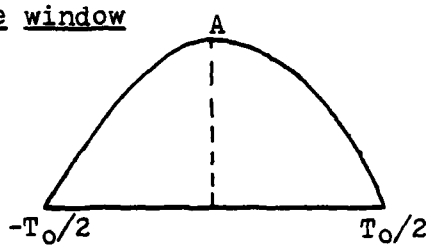


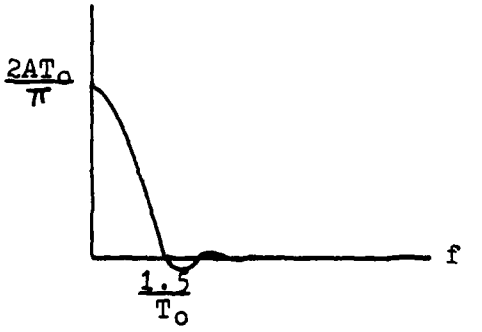


$$w(t) = A\left[1 - \frac{|t|}{T_o/2}\right] \quad |t| \leq T_o/2$$
$$= 0 \quad \text{Otherwise}$$

$$W(f) = \frac{AT_o}{2}\left(\frac{\sin(\frac{\pi}{2} fT_o)}{\frac{\pi}{2} fT_o}\right)^2$$

Cosine window

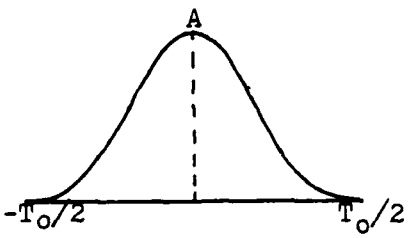


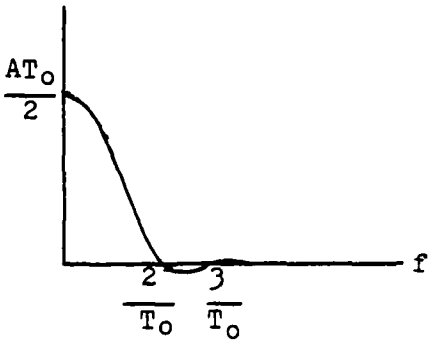


$$w(t) = A \cos \frac{\pi t}{T_o} \quad |t| \leq \frac{T_o}{2}$$
$$= 0 \quad \text{otherwise}$$

$$W(f) = \frac{AT_o}{\pi}\left(\frac{\cos \pi fT_o}{1-(2\,fT_o)^2}\right)$$

Hanning window

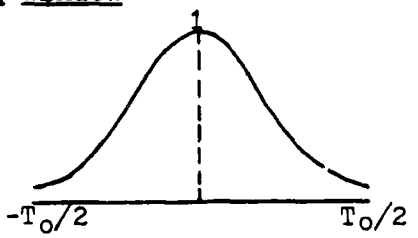


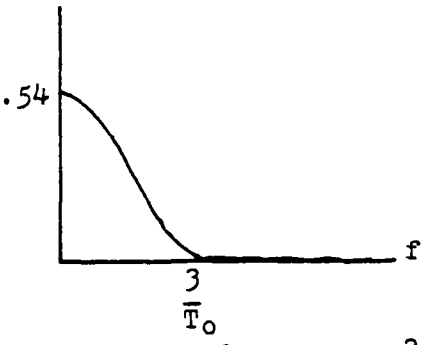


$$w(t) = \frac{A}{2}\left[1+\cos\left(\frac{2\pi t}{T_o}\right)\right] \quad |t| \le \frac{T_o}{2}$$
$$= 0 \quad \text{otherwise}$$

$$W(f) = \frac{AT_o}{2}\,\frac{\sin(\pi f T_o)}{(\pi f T_o)(1-(fT_o)^2)}$$

Hamming window

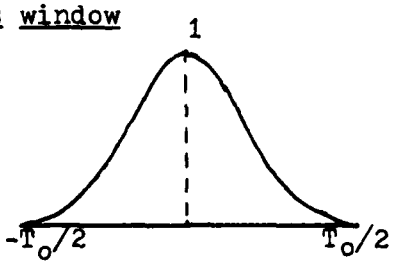


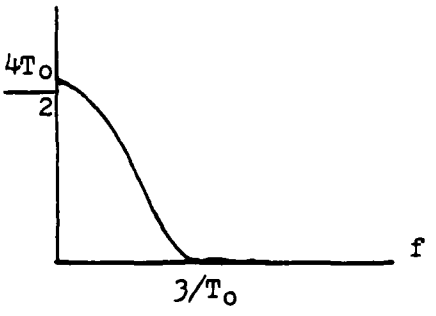


$$w(t) = .54+.46\cos(2\pi t/T_o) \quad |t| \le T_o/2$$
$$= 0 \quad \text{otherwise}$$

$$W(f) = \frac{(.54\pi^2-.08(\pi f T_o)^2)\sin(\pi f T_o)}{\pi f T_o(\pi^2-(\pi f T_o)^2)}$$

Papoulis window

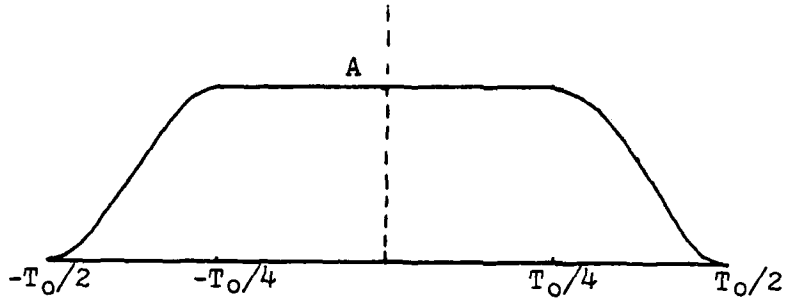




$$w(t) = \frac{1}{\pi}\left|\sin\frac{2\pi t}{T_o}\right| + \left(1-\frac{|t|}{T_o/2}\right)\cos\left(\frac{2\pi t}{T_o}\right) \quad |t| \le T_o/2$$

$$W(f) = 2\pi^2 T_o \quad \frac{1+\cos(\pi f T_o)}{((\pi f T_o)^2-\pi^2)^2}$$

Tukey window



$$w(t) = \begin{cases} = \frac{A}{2}\left[1 - \cos\frac{2\pi(t + T_o/2)}{T_o/2}\right], & \frac{-T_o}{2} \le t \le \frac{-T_o}{4} \\ = A & \frac{-T_o}{4} \le t \le \frac{T_o}{4} \\ = \frac{A}{2}\left[1 - \cos\frac{2\pi(T_o/2 - t)}{T_o/2}\right] & \frac{T_o}{4} \le t \le \frac{T_o}{2} \\ = 0 & \text{otherwise} \end{cases}$$

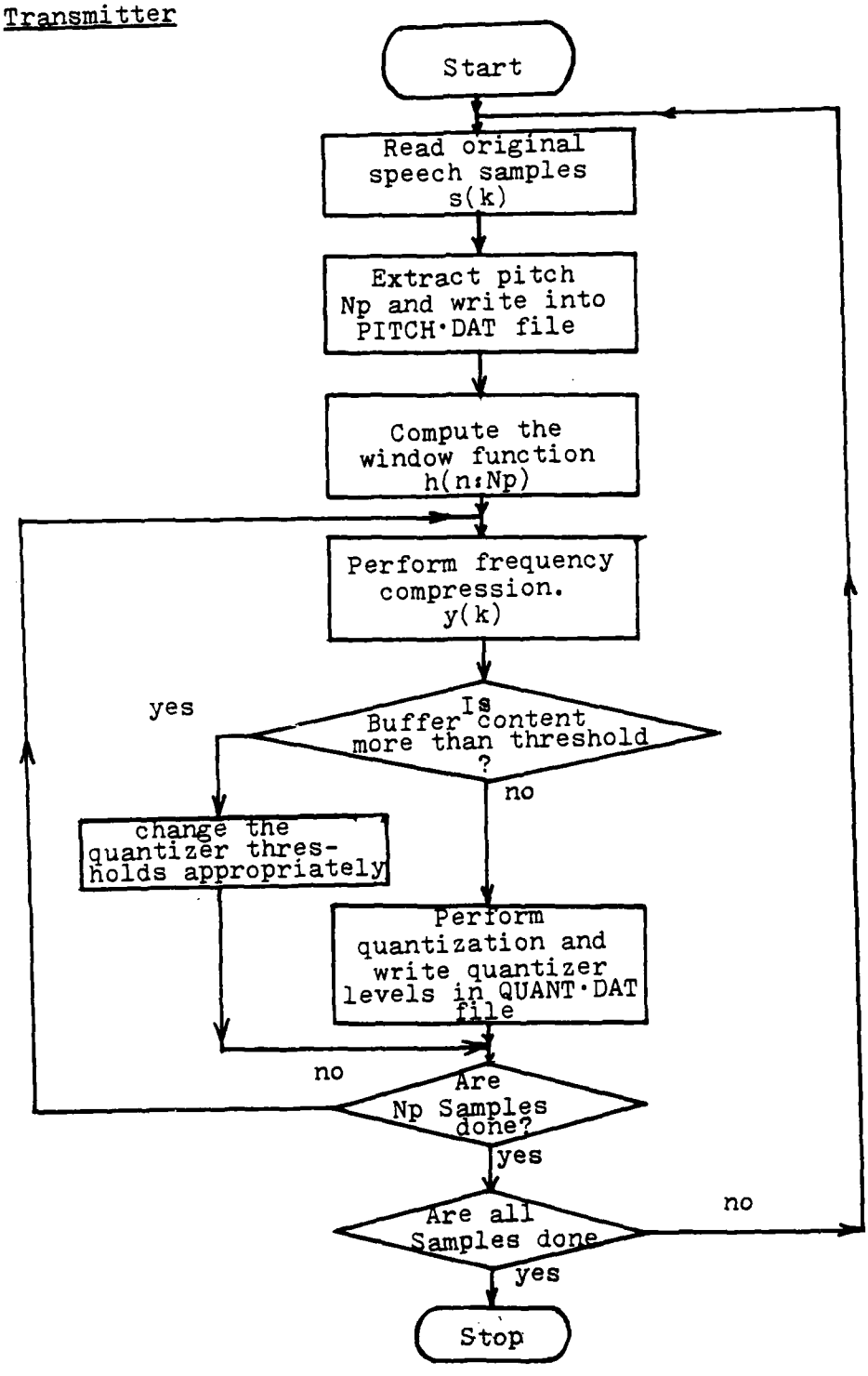
$$W(f) = \frac{T_o}{4}\,\frac{\sin\left(\frac{2\pi f T_o}{4}\right)}{\frac{2\pi f T_o}{4}} + \frac{T_o}{2}\,\frac{\sin\left(\frac{2\pi f T_o}{2}\right)}{\frac{2\pi f T_o}{2}}$$

$$- \cos\left(\frac{2\pi f T_o}{8}\right)\,\sin\left(\frac{2\pi f T_o 3}{8}\right)\cdot\frac{2(2\pi f)}{((2\pi f)^2 - (\frac{2\pi}{T_o/2})^2)}$$

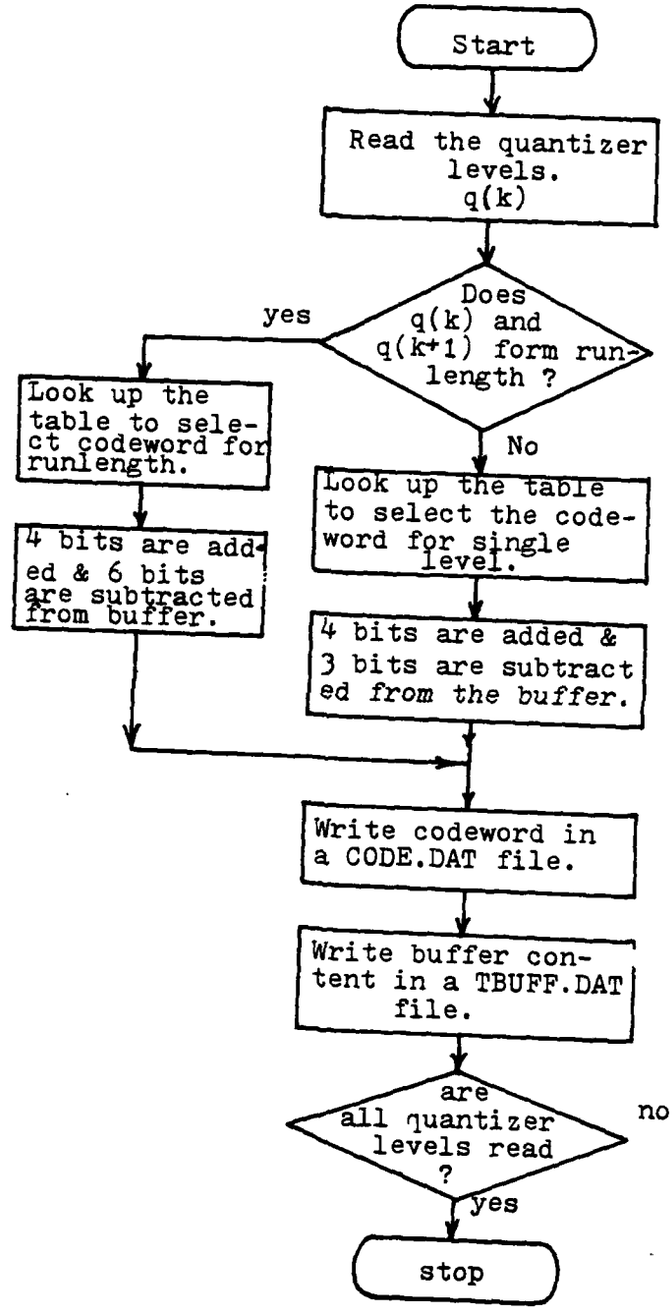
APPENDIX B

FLOW CHARTS

Transmitter

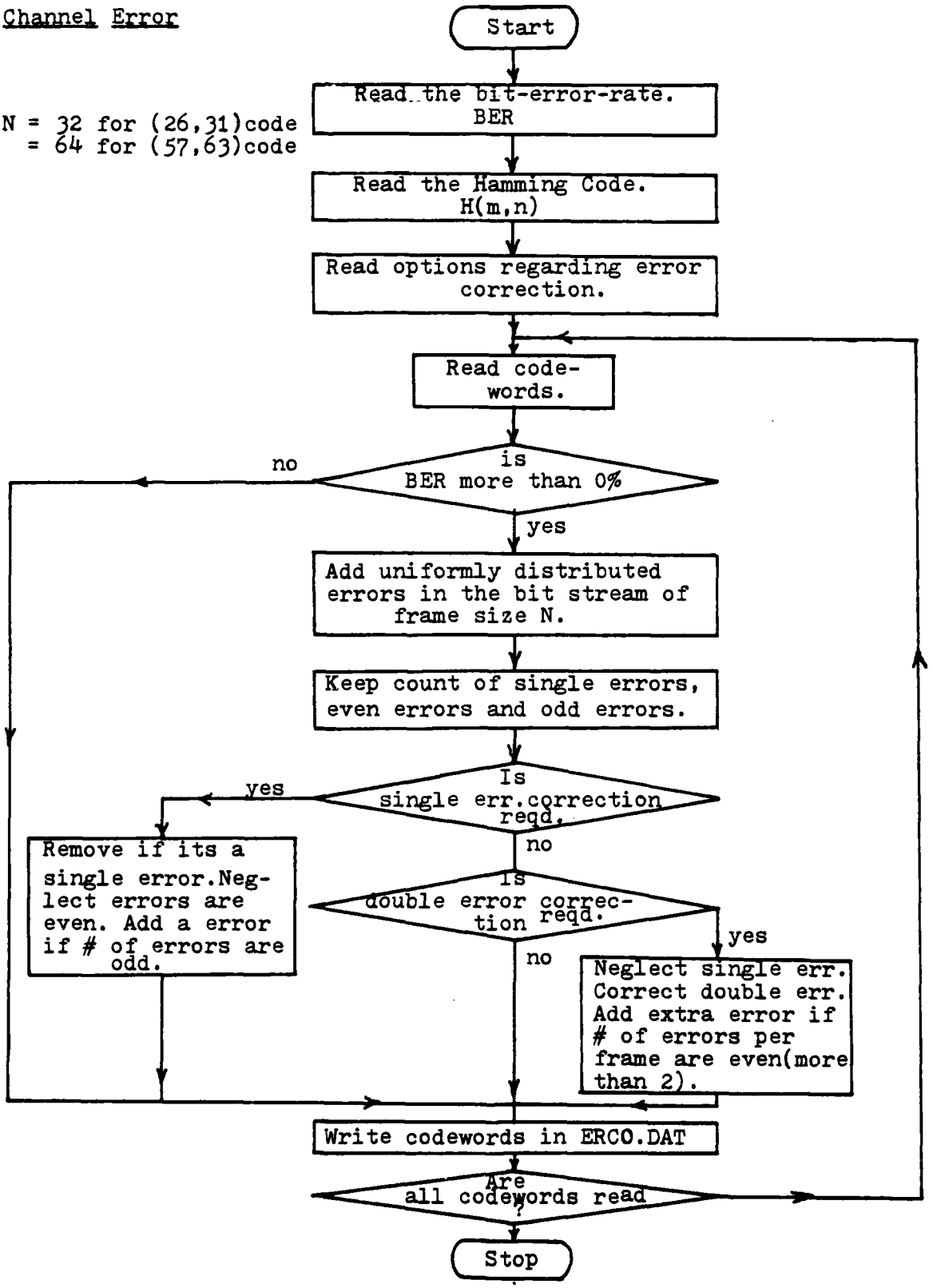


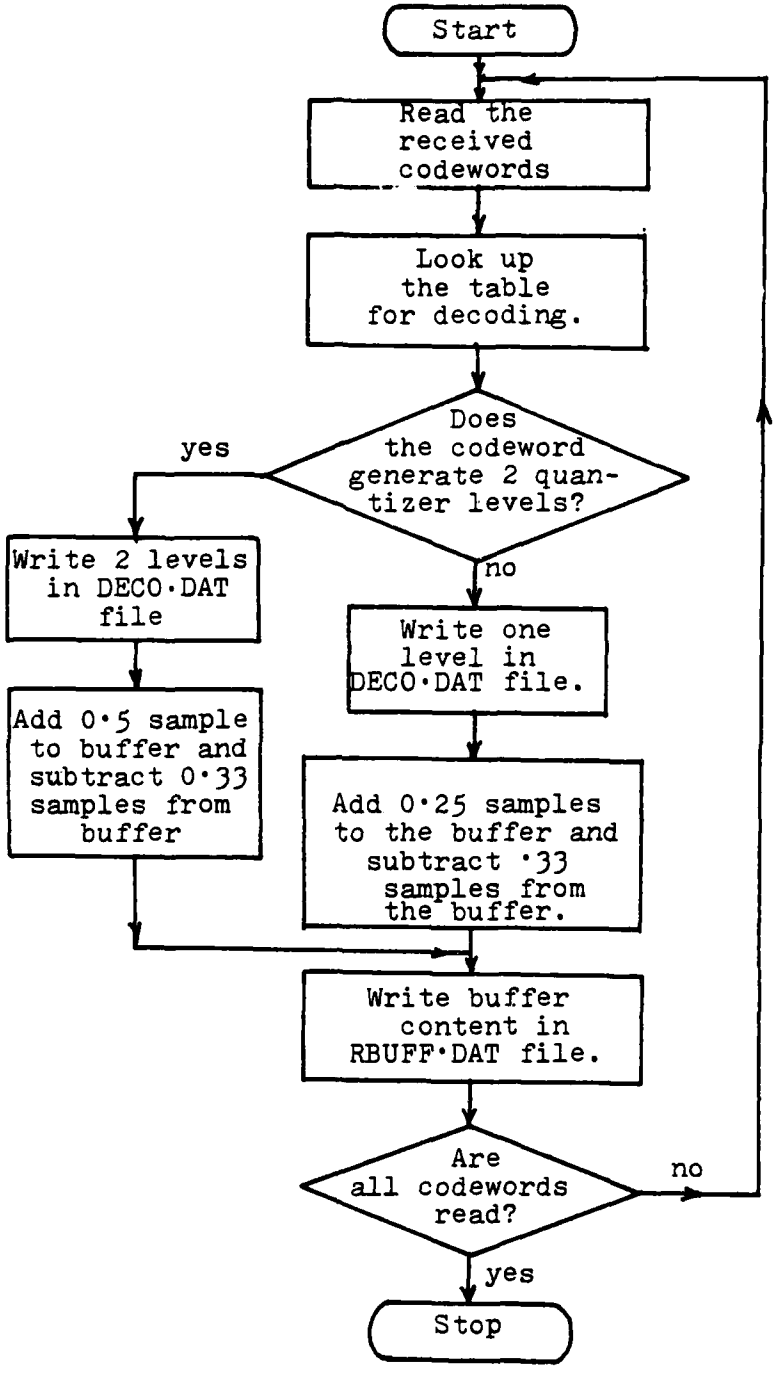
Encoder


Start
Read the quantizer levels. q(k)
Does q(k) and q(k+1) form run-length ?
yes
No
Look up the table to select codeword for runlength.
4 bits are added & 6 bits are subtracted from buffer.
Look up the table to select the codeword for single level.
4 bits are added & 3 bits are subtracted from the buffer.
Write codeword in a CODE.DAT file.
Write buffer content in a TBUFF.DAT file.
are all quantizer levels read ?
no
yes
stop

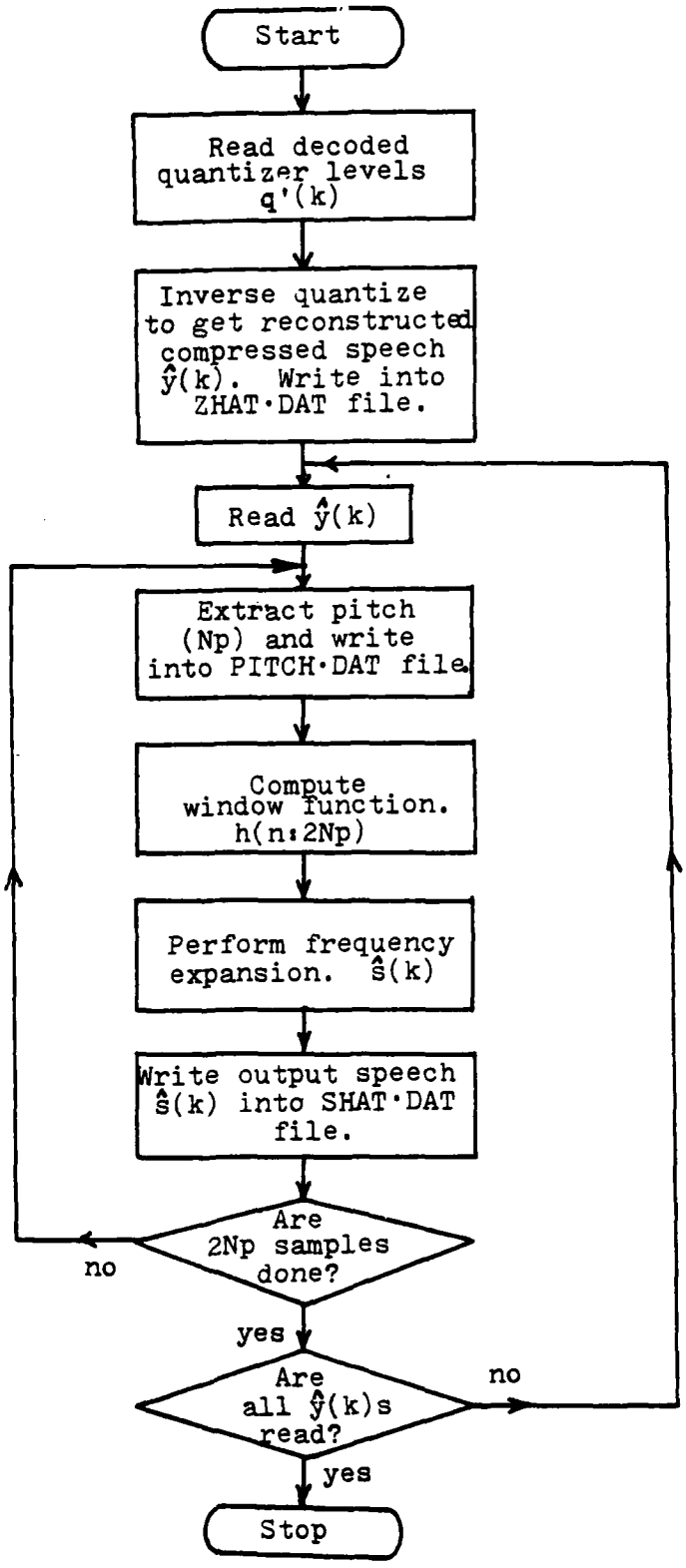
Channel Error

N = 32 for (26,31)code
= 64 for (57,63)code




Start
Read the received codewords
Look up the table for decoding.
Does the codeword generate 2 quantizer levels?
yes
no
Write 2 levels in DECO·DAT file
Add 0·5 sample to buffer and subtract 0·33 samples from buffer
Write one level in DECO·DAT file.
Add 0·25 samples to the buffer and subtract ·33 samples from the buffer.
Write buffer content in RBUFF·DAT file.
Are all codewords read?
no
yes
Stop

Receiver


Start
Read decoded quantizer levels q'(k)
Inverse quantize to get reconstructed compressed speech ŷ(k). Write into ZHAT·DAT file.
Read ŷ(k)
Extract pitch (Np) and write into PITCH·DAT file.
Compute window function. h(n:2Np)
Perform frequency expansion. ŝ(k)
Write output speech ŝ(k) into SHAT·DAT file.
Are 2Np samples done?
no
yes
Are all ŷ(k)s read?
no
yes
Stop

APPENDIX C

USER'S GUIDE AND SOURCE LISTINGS

The system software developed for the speech coder for the bit of 9.6 kb/s, consists of six program modules. They are as follows.

1. Options (OPTION•FTN): This program asks for various options required for transmitter and receiver programs and create OPT1•DAT and OPT2•DAT files to be used by the transmitter and the receiver program respectively.

2. Transmitter (TRAN•FTN): This combines the frequency compression and the quantization using Adaptive Residual Coder. The performance and the statistics are printed on the unit 6. The program asks for headers to various files and produces the quantizer level file (QUANT•DAT).

3. Encoder (ENCO•FTN): This program reads the quantizer levels and generates the codewords which are written into CODE•DAT file. The transmitter buffer (TBUFF•DAT) is also simulated.

4. Channel Error (CHER5763•DAT and CHER2631•FTN): There are two modules to simulate noisy channel with the error correction simulation incorporated in them. CHER5763•FTN uses [57,63] Hamming Code and the other module uses [26,31] Hamming Code. These programs ask for the percentage BER, the type of error correction used and output the number of errors per frame. The residual errors and the BER after error correction are also displayed on the terminal.

5. Decoder (DECO•FTN): Reads the corrupted and/or corrected code words from ERCO•DAT file and decodes them and puts them into DECO•DAT file. The number of samples added or deleted due to channel error can be found by

noting the number of samples in the tail (which is displayed on the terminal) with and without error. Note that this number is modulo 256. The receiver buffer (RBUFF•DAT) is also simulated in this program.

6. Receiver (RCVR•FTN): reads the received quantizer levels, performs inverse quantization, (ZHAT•DAT) does frequency expansion and writes the speech output in the SHAT•DAT file. This program brings all the quantizer levels in the memory simultaneously. Hence, this program will give error for sentences longer than 15600 samples. The longer sentences could be processed by increasing the Q buffer array size.

To run these modules Indirect Command file is written. The task files of the above modules and the ARC parameter file should exist in the same UIC. If the task files are not available, following switches should be used to build them.

```
>ACTFIL = 10
>UNITS  = 10
>ASG    = SY:6
```

The indirect command file asks options regarding the various files to be retained for further use or to be deleted.

```
;       ****************************************************************
;       *                                                              *
;       *              INDIRECT COMMAND FILE  FOR                      *
;       *                                                              *
;       *       9.6 KB/S  T D H S - A R C    S Y S T E M               *
;       *                                                              *
;       ****************************************************************
;
;       DATE          : JUNE 30,1981
;       NAME          : ARUN K. PANDE
;
;       THIS PROGRAM CONSISTS OF EXECUTION OF FIVE MODULES. THESE MODULES ARE
;
;                                1. TRANSMITTER        TRAN.FTN
;                                2. ENCODER            ENCO.FTN
;                                3. CHANNEL            CHER5763.FTN
;                                   (+ ERROR CORR)     CHER2631.FTN
;                                4. DECODER            DECO.FTN
;                                5. RECEIVER           RCVR.FTN
;
;       TO RUN ABOVE INDIRECT COMMAND FILE, IT IS ASSUMED THAT THE TASKS OF
;       ABOVE MODULES EXISTS IN THE SAME UIC,. IF THEY DONOT, BUILD THE TASKS
;       USING FOLLOWING SWITCHES COMMON TO ALL MODULES.
;
;                                >ACTFIL=10
;                                >UNITS=10
;                                >ASG=DK1:6
;
        .ENABLE SUBSTITUTION
        .SETS S1 "QUANT.DAT;*,CODE.DAT;*,ERCO.DAT;*,DECO.DAT;*,ZHAT.DAT;*"
        .SETS S2 "FOR006.DAT;*"
        .SETS S3 "TBUF.DAT;*,RBUF.DAT;*"
        RUN OPTION
        .WAIT
        RUN TRAN
        .ASK DOPT1 DO YOU WANT TO DELETE TRANSMITTER OPTION FILE
        .IFF DOPT1 .GOTO 10
        .WAIT PIP
        PIP OPT1.DAT;*/DE
.10:    .WAIT
        RUN ENCO
.20:    .ASK A1 DO YOU WANT TO USE [57,63] HAMMING CODE
        .IFT A1 .GOTO 30
        .ASK A2 DO YOU WANT TO USE [26,31] HAMMING CODE
        .IFT A2 .GOTO 40
        .GOTO 20
.30:    .WAIT
        RUN CHER5763
        .GOTO 50
.40:    .WAIT
        RUN CHER2631
.50:    .WAIT
        RUN DECO
        .WAIT
        RUN RCVR
        .WAIT PIP
        PIP 'S1'/DE
        .ASK CHEK DO YOU WANT TO KEEP STATISTICS AND PITCH DATA
        .IFT CHEK .GOTO 60
        .WAIT PIP
        PIP 'S2'/DE
.60:    .WAIT
        .ASK CHEKB DO YOU WANT TO KEEP TRAN AND RCVR BUFFER FILES
        .IFT CHEKB .GOTO 70
        .WAIT PIP
        PIP 'S3'/DE
.70:    .DELAY 01S
        .WAIT PIP
        PIP RECSAM.DAT;*,NEWBLO.DAT;*/DE
```

```
C       THIS PROGRAM CREATES THE FILES FOR OPTIONS REQUUIRED IN TRANSMITTER
C       AND RECEIVER PROGRAMS.
C
C
        INTEGER*2 FNAME(16),IBT(3)
        DIMENSION GAMA(3)
C
C
        OPEN(UNIT=1,TYPE='NEW',NAME='OPT1.DAT',CARRIAGECONTROL='LIST')
C
C
        TYPE *,' ENTER THE ORIGINAL SPEECH FILENAME:( < 32 CHAR )'
        ACCEPT 10,FNAME
10      FORMAT(16A2)
        WRITE(1,10)FNAME
C
C
        TYPE *,' ENTER CLIPPING LEVEL FOR PITCH EXTRACTION.'
        TYPE *,' [ 1.0= 100% CLIPPING; 0.0= NO CLIPPING]'
        TYPE *,' TYPICAL VALUE: 0.3 TO 0.6 '
        ACCEPT *,CLPP
        WRITE(1,*)CLPP
C
        TYPE *,' ENTER BLOCK LENGTH (KBLK), SEARCHING RANGE (ITMIN,ITMAX)'
        TYPE *,' FOR EXAMPLE: KBLK=80, ITMIN=20, ITMAX=100'
        ACCEPT *,KBLK,ITMIN,ITMAX
        WRITE(1,*)KBLK,ITMIN,ITMAX
C
        TYPE *,' ENTER ARC PARAMETER FILENAME: [ < 32 CHAR ]'
        ACCEPT 10,FNAME
        WRITE(1,10)FNAME
C
C
        TYPE *,' BLOCKLENGTH IN ARC TO CALCULATE SEGSNR, TYPICAL: 64'
        ACCEPT *,NBL
        WRITE(1,*)NBL
C
        TYPE *,' SELECT THE TYPE OF PITCH ESTIMATOR:'
        TYPE *,' 1 = AUTOCORRELATION;  2 = AMDF       ; 3 = CLIPPED SP AUTO'
        TYPE *,' 4 = CLIPPED SP AMDF;  5 = 3-VALUE C CLIPPED AUTOCORRELATION'
        TYPE *,' 6 = 3-VALUE C CLIPPED SPEECH AMDF; 7 = 2-VALUE C CLIPPED '
        TYPE *,' SPEECH AUTOCORRELATION; 8 = 2-VALUE C CLIPPED AMDF'
        ACCEPT *,IOPT
        WRITE(1,*)IOPT
C
        TYPE *,' DO YOU WANT TO HAVE A BUFFER CONTROL? [ 1=YES; 2=NO ]'
        ACCEPT *,IBCNT
        WRITE(1,*)IBCNT
C
        IF(IBCNT .NE. 1)GOTO 20
        TYPE *,' ENTER BUFFER THRESHOLDS FOR BUFFER CONTROL:'
        TYPE *,' FOR EXAMPLE: [ 600, 800, 900 ]'
        ACCEPT *,IBT
        WRITE(1,*)IBT
        TYPE *,' ENTER THE SCALARS TO CHANGE THE QUANTIZER THRESHOLDS'
        TYPE *,' FOR EXAMPLE: [ 0.5, 0.7, 1.0]'
        ACCEPT *,GAMA
        WRITE(1,*)GAMA
20      CONTINUE
C
        TYPE *,' ENTER HAMMING ERROR CORRECTING CODE;'
        TYPE *,' ( INFO BITS, TOTAT NUMBER OF BITS ); FOR EXAMPLE:(57,63)'
        ACCEPT *,INFO,NTOT
        WRITE(1,*)INFO,NTOT
```

```
C
C
C
C       OPTIONS FOR THE RECEIVER PROGRAM.
C
        OPEN(UNIT=2,NAME='OPT2.DAT',TYPE='NEW',CARRIAGECONTROL='LIST')
C
        WRITE(2,10)FNAME
C
        TYPE *,' ENTER THE CLIPPING LEVEL FOR PITCH EXTRACTION AT RECEIVER:'
        TYPE *,' [ 1.0 - 100% CLIPPING; 0.0 - NO CLIPPING ]'
        TYPE *,' TYPICAL VALUE= 0.3 TO 0.6'
        ACCEPT *,CLP
        WRITE(2,*)CLP
C
        TYPE *,' SELECT THE TYPE OF PITCH ESTIMATOR:'
        TYPE *,' 1 = AUTOCORR;          2 = AMDF;       3 = CLPPD SP AUTOCORR'
        TYPE *,' 4 = CLPPD SP AMDF;     5 = 3-VALUE CENTER CLIPPED AUTOCORR'
        TYPE *,' 6 = 3-VAL C CLPPD AMDF; 7 = 2-VALUE CENTER CLIPPED AUTOCORR'
        TYPE *,' 8 = 2-VALUE CENTER CLIPPED AMDF'
        ACCEPT *,IOPT1
        WRITE(2,*)IOPT1
C
        TYPE *,' ENTER BLOCK LENGTH (KBLK), SEARCHING RANGE (ITMIN,ITMAX)'
        TYPE *,' TYPICAL VALUES: 40, 20,100'
        ACCEPT *,KBLK,ITMIN,ITMAX
        WRITE(2,*)KBLK,ITMIN,ITMAX
C
        STOP
        END
```

```
C       ********************************************************
C       *                                                      *
C       *             T D H S  -  A R C   S Y S T E M          *
C       *                   T R A S M I T T E R                *
C       *                                                      *
C       *                            BY ARUN K. PANDE          *
C       ********************************************************
C
C       THIS IS A NEW APPROACH TO SPEECH DIGITIZATION AT MEDIUM BAND BIT
C       RATES OF 9.6 TO 16 KB/S. THE TECHNIQUE IS BASED ON A COMBINATION
C       OF TIME-DOMAIN HARMONIC SCALING ( TDHS ) AND ADAPTIVE RESIDUAL
C       CODER (ARC).
C       TDHS ALGORITHM CONSISTS OF PROPERLY WEIGHTING SEVERAL ADJSCENT
C       INPUT SIGNAL SEGMENT OF PITCH DEPENDENT DURATION BY SUITABLE
C       WINDOW FUNCTIONS. AS A RESULT OF THIS, THE NUMBER OF SAMPLES
C       TO BE TRANSMITTED CAN BE REDUCED BY A FACTOR OF TWO. IF THE BIT
C       RATE IS KEPT THE SAME, THE NUMBER OF BITS ALLOWED PER SAMPLE IS
C       DOUBLED, AND THE PERFORMANCE OF THE CODER CAN BE IMPROVED
C       SIGNIFICANTLY. WITH THE MORE SLOWLY VARYING FREQUENCY DIVIDED
C       SIGNAL AS INPUT, THE PREDICTION AND ASSOCIATED QUANTIZATION IN THE
C       ARC SYSTEM WILL PERFORM BETTER, THUS INCREASING THE PERFORMANCE
C       OF THE SYSTEM.
C
C       PROGRAM NAME:  TRAN.FTN
C       DATE         :  JUNE 30,1981
C
C
C       THIS PROGRAM READS  FOLLOWING OPTIONS:
C                              1. BLOCK LENGTH (KBLK),SEARCHING RANGE(ITMIN
C                                                                    ,ITMAX)
C                              2. NAME OF THE SPEECH FILE.
C                              3. NAME OF THE PARAMETER FILE.
C                              4. HEADERS FOR QUANTIZER OUTPUT
C                              5. OPTION REGARDING PITCH METHOD TO BE USED.
C                              6. BLOCK LENGTH NBL FOR SEGSNR.
C                              7. CLIPPING LAVEL TO GENERATE CLIPPED SPEECH
C                              8. BUFFER CONTROL OPTIONS.
C                              9. HAMMING CODE TO BE USED.
C                             10. HEADER FOR PITCH PRINTOUT.
C
      INTEGER HD(40),IPICH(400),FNAME(16),SQ,Q,FILEN
      INTEGER*2 SPEECH(400),FBCNT,OLDQ
      DIMENSION Y(256), H(400), SQ(16),IBUFF1(512),IBUFF2(512)
      INTEGER*2 F1,F2,F3,F4,STAT
      COMMON /PRED/G,ND,RMSMIN,ALP,AINV,KQ,NSPSAM,A(12),DVHAT(12),EV,
     1 ISTAT1(40),EP,ALAD,SPERB1(200),SPERB2(200),V(12),SNRB(200)
     2 ,SNRQB(200)
      COMMON /RMS/RMS,NBL,IARG,ENGY1,ENGY2,ENGY3,ENGY4,SPER1,SPER2
      COMMON /CLIPP/CLPP
      COMMON /ADDN/F1,F2,F3,F4,BT1,BT2,BT3
     1 ,JCNT,STAT(30),GAMA(4)
      COMMON /FN/FILEN(16)
C
      MODN(K) = K - (K-1)/16 * 16
C
      OPEN(UNIT=3,TYPE='OLD',NAME='OPT1.DAT')        !OPEN THE OPTION FILE.
C
C------ OPEN THE SPEECH FILE; READ THE HEADER AND THE SPEECH.
C
      READ(3,10)FNAME
10    FORMAT(16A2)
      OPEN(UNIT=1, TYPE='OLD', READONLY, NAME=FNAME, SHARED)! OPEN SPEECH FILE
      READ(1,20)NSENT,IRATE,NSAMP,IUPPR,ILOWR,NTERMS,HD
20    FORMAT(6I5,10X,40A1)
```

```
40      CONTINUE
C
C------ CREATE NEW FILE FOR QUANTIZER OUTPUT. WRITE
C------ THE HEADER ON QUANTIZER FILE.
C
        OPEN(UNIT=2, TYPE='NEW', NAME='QUANT.DAT',CARRIAGECONTROL='LIST')
        TYPE *,' TYPE THE HEADER FOR THE QUANTIZER FILE:'
        ACCEPT 50, HD
50      FORMAT(40A1)
        NSPSAM = NSAMP/2
        WRITE(2,20)NSENT,IRATE,NSPSAM,IUPPR,ILOWR,NTERMS,HD
C
C
        READ(3,*)CLPP
C
C------ DEFINE AND INITIALIZE VARIOUS PARAMETERS.
C
        NUMP = 0
        READ(3,*)KBLK1,ITMIN,ITMAX          ! BLOCK LENGTH AND SEARCHING RANGE
        IOFF  = 0
        ICNT  = 0
        BUFCNT=0.
        JJ    = 1
        NADD  = 0
        F1    = 0
        F2    = 0
        F3    = 0
        F4    = 0
        JCNT  = 1
        GAMA(1)=1.0
        NUM1  = KBLK1 + ITMAX
        READ(3,10)FILEN
        FILEN(32)=0
        CALL INSTRT
C
C
        READ(3,*)IOPT2              ! TYPE OF PITCH ESTIMATOR.
        READ(3,*)FBCNT              ! BUFFER CONTROL FLAG.
        IF(FBCNT .NE. 1)GOTO 155
        READ(3,*)BT1,BT2,BT3        ! BUFFER THRESHOLDS.
        READ(3,*)GAMA1,GAMA2,GAMA3      ! SCALARS TO CHANGE QUANT THRESHOLDS
        GAMA(2)=GAMA1
        GAMA(3)=GAMA2
        GAMA(4)=GAMA3
        GOTO 158
155     GAMA(2)=1.0
        GAMA(3)=1.0
        GAMA(4)=1.0
        BT1=3000.0
        BT2=4500.
        BT3=6000.
158     CONTINUE
        READ(3,*)NUMQ1,NQPAR            ! HAMMING CODE.
        NPARI=NQPAR-NUMQ1
        AVGBIT=(FLOAT(NUMQ1)/FLOAT(NQPAR))*3.0
C
C
C------ TRANFER MAXIMUM LAG (ITMAX) PLUS BLOCKSIZE (KBLK1) SPEECH SAMPLES
C------ TO BUFFER1.
C       CHECK IF NUM1 IS EXACT MULTIPLE OF 16.
        IF(MOD(NUM1,16) .EQ. 0)GOTO 55
C       IF NOT, MAKE IT EXACT MULTIPLE OF 16.
              NUM1=(NUM1/16)*16+16
55      READ(1,30)(IBUFF1(I), I=1,NUM1)
```

```
        ICNT=ICNT+NUM1
C
C       EXTRACT PITCH FROM SAMPLES IN BUFFER1.
C
        CALL PITCH(IBUFF1, NP, ITMAX, ITMIN,  KBLK1, NUMP, NUM1, IOPT2)
        IPICH(JJ) = NP
        NP2 = NP + NP
C
56      CONTINUE
C
C------ CHECK IF NUMBER OF SAMPLES IN BUFFER1 IS LESS THAN TWO TIMES PITCH
C------ PERIOD SAMPLES; IF LESS, ADD TO IT FROM SPEECH BUFFER AND THEN
C------ PROCEED TO FREQUENCY DIVISION.
        IF(NUM1 .LT. NP2)GOTO 61
C       COMPUTE HOW MANY EXTRA SAMPLES IBUFF1 HAS.
                NUM12 = NUM1 - NP2
C       IS NUM1 EXACTLY EQUAL TO NP2?
C
                IF(NUM12 .EQ. Ø)GOTO 73
C
C       IF NOT, TRANSFER EXTRA SAMPLES FROM IBUFF1 TO IBUFF2.
C
                DO 58   I=1,NUM12
                        IBUFF2(I) = IBUFF1(NP2+I)
58              CONTINUE
C
        GOTO 73
C
C       COMPUTE HOW MANY MORE SAMPLES ARE NEEDED IN IBUFF1 TO MAKE THE
C       NUMBER EXACTLY EQUAL TO 2*NP.
C
61              MORE = NP2 - NUM1
C
C       IS MORE EXACT MULTIPLE OF 16.
C
                IF(MOD(MORE,16) .EQ. Ø)GOTO 7Ø
C
C       IF NOT, FIND THE NUMBER WHICH EXACT MULTIPLE OF 16 AND IS CLOSEST TO
C       MORE.
C
                NUM2 = (MORE/16)*16 + 16
                READ(1,3Ø,END=533)(SPEECH(I),I=1,NUM2)
533             CONTINUE
                ICNT = ICNT + NUM2
                IF(ICNT .GT. NSAMP)GOTO 999
                NUM3 = NUM2 - MORE
                DO 64   I=1,MORE
                IBUFF1(NUM1+I) = SPEECH(I)
64              CONTINUE
                DO 67   I=1,NUM3
                        IBUFF2(I) = SPEECH(MORE+I)
67              CONTINUE
                NUM12 = NUM3
                GOTO 73
C
C       SINCE "MORE" IS EXACT MULTIPLE OF 16, READ "MORE" SAMPLES FROM
C       SPEECH FILE AND PUT IN IBUFF1.
C
7Ø              READ(1,3Ø,END=537)(IBUFF1(NUM1+I), I=1,MORE)
537             CONTINUE
                ICNT = ICNT + MORE
                IF(ICNT .GT. NSAMP)GOTO 999
                NUM12=Ø
C
```

```
C
C       CALCULATE THE WINDOW FUNCTION.
C
73              CALL WINDOW(H,NP)                   ! COMPUTE TRIANGULAR WINDOW.
C
C
C------ PERFORM FREQUENCY DIVISION OPERATION.
C
        DO 72   I = 1, NP                   ! FREQUENCY DIVISION OPERATION.
                NUM4 = NP +  I
                Y(I) = FLOAT(IBUFF1(NUM4)) + H(I) * FLOAT(IBUFF1(I) -
     1                    IBUFF1(NUM4))
                IARG = NADD + I
                YY = Y(I)
C------ CALL ARC SUBROUTINE WHICH RETURNS THE QUANTIZER  OUTPUT CORRESPONDING
C------ TO FREQUENCY DIVIDED SPEECH SAMPLE AND ALSO NOISY SPEECH SAMPLE.
                CALL ARC(YY,Q,YHAT)
                JQQ=Q
                CALL BUFCTL(BUFCNT,JQQ,AVGBIT)  ! BUFFER CONTROL IF OVERFLOW.
C------ WRITE THE QUANTIZER OUTPUT IN THE FILE.
                N1 = MODN(IARG)
                SQ(N1) = Q
                IF(N1 .EQ. 16) WRITE(2,33) SQ
72      CONTINUE
C
C
        NADD = NADD + NP
        NUM1 = KBLK1 + ITMAX
C
C       THERE ARE ALREADY NUM12 SAMPLES IN IBUFF2. PUT (NUM1-NUM12) MORE
C       SAMPLES IN IT.
C
                NSP = NUM1 - NUM12
C
C       MAKE NSP EXACT MULTIPLE OF 16.
                NSP = (NSP/16) * 16 + 16
                READ(1,30,END=541)(IBUFF2(NUM12+I), I=1,NSP)
541             CONTINUE
                ICNT=ICNT+NSP
                IF(ICNT .GT. NSAMP)GOTO 999
C
        NUMNSP = NUM12 + NSP
C
C       EXTRACT PITCH
C
C
C
        JJ = JJ + 1
        CALL PITCH(IBUFF2,NP,ITMAX,ITMIN,KBLK1, NUMP,NUMNSP,IOPT2)
        IPICH(JJ) = NP
        NP2 = NP + NP
C
C       DOES IBUFF2 HAVE SAMPLES LESS THAN 2*NP.
C
                IF(NUMNSP .LT. NP2)GOTO 85
C
C       HOW MANY EXTRA SAMPLES IBUFF2 HAS?
                NUM12 = NUMNSP - NP2
C
C       DOES IBUFF2 HAVE EXACT 2*NP SAMPLES?
C
                IF(NUM12 .EQ. 0)GOTO 82
C
C       TRANSFER EXTRA SAMPLES TO IBUFF1.
```

```
C
                DO 79    I=1,NUM12
                         IBUFF1(I)=IBUFF2(NP2+I)
79              CONTINUE
82        GOTO 88
C
C         HOW MANY MORE SAMPLES ARE TO BE ADDED TO IBUFF2 SO THAT IT WILL
C         HAVE 2*NP SAMPLES ?
C
85              MORE = NP2 - NUMNSP
C
C         IS MORE EXACT MULTIPLES OF 16?
C
                IF(MOD(MORE,16) .EQ. Ø)GOTO 91
C
C         IF NOT, MAKE IT EXACT MULTIPLE OF 16.
C
                NUM2=(MORE/16) * 16 + 16
                READ(1,3Ø,END=544)(SPEECH(I),I=1,NUM2)
544             CONTINUE
                ICNT=ICNT+NUM2
                IF(ICNT .GT. NSAMP)GOTO 999
C
C         HOW MANY EXTRA SAMPLES ARE READ THAN NEEDED.
C
                NUM3 = NUM2 - MORE
C
C         TRANSFER "MORE" SAMPLES TO IBUFF2 AND NUM3 SAMPLES TO IBUFF1.
C
                DO 86 I=1,MORE
                         IBUFF2(NUMNSP+I)=SPEECH(I)
86              CONTINUE
                DO 87    I=1,NUM3
                         IBUFF1(I)=SPEECH(MORE+I)
87              CONTINUE
C
          NUM12=NUM3
          GOTO 88
91        READ(1,3Ø,END=547)(IBUFF2(I+NUMNSP),I=1,MORE)
547       CONTINUE
          ICNT=ICNT+MORE
          NUM12=Ø
88        CALL WINDOW(H,NP)                ! COMPUTE TRIANGULAR WINDOW FUNCTION.
C
C
C
C
          DO 92    I = 1, NP               ! FREQUENCY DIVISION OPERATION.
                   NUM4 = NP +  I
                   Y(I) = FLOAT( IBUFF2(NUM4)) + H(I) * FLOAT( IBUFF2(I) -
     1                        IBUFF2(NUM4))
                   IARG = NADD + I
                   YY = Y(I)
C------   CALL SUBROUTINE ARC.
                   CALL ARC(YY,Q,YHAT)     ! QUANTIZE COMPRESSED SP.
                   JQQ=Q
                   CALL BUFCTL(BUFCNT,JQQ,AVGBIT)  ! CONTROL BUFFER IF OVERFLOW.
                   N1 = MODN(IARG)
                   SQ(N1) = Q
                   IF(N1 .EQ. 16 ) WRITE(2,33)SQ
92        CONTINUE
          NADD = NADD + NP
C
C         THERE ARE ALREADY NUM12 SAMPLES IN IBUFF1.TRANSFER NSP=NUM1-NUM12
```

```
C       SAMPLES TO IT.
C
              NSP = NUM1 - NUM12
              NSP = (NSP/16)*16 + 16
              READ(1,30,END=551)(IBUFF1(NUM12+I),I=1,NSP)
551           CONTINUE
        ICNT=ICNT+NSP
        IF(ICNT.GT.NSAMP)GOTO 999
        NUMNSP=NUM12+NSP
C
C
        JJ = JJ + 1
C
        CALL PITCH( IBUFF1, NP, ITMAX, ITMIN,  KBLK1, NUMP,NUMNSP,IOPT2)
        IPICH(JJ) = NP
        NP2 = NP + NP
        NUM1=NUMNSP
        GOTO 56
999     CONTINUE
C
C
        CALL INEND              ! PRINTOUT ON UNIT 6 ALL THE STATISTICS.
        TYPE *,' TYPE THE HEADER FOR THE PRINTOUT.(40CHAR ONLY)'
        ACCEPT 775,HD
775     FORMAT(40A1)
        WRITE(6,774)HD
774     FORMAT(///,40A1,//)
        WRITE(6,776)
776     FORMAT(////6X,' SAMPLE NUMBER ',6X,' PITCH PERIOD '//)
        ISTRT = 1
        IEND  = KBLK1+ITMAX
        DO 777  I = 1,NUMP
              IP = IPICH(I)
              WRITE(6,778)ISTRT,IEND,IP
              I2P = IP + IP
              ISTRT = ISTRT + I2P
              IEND  = IEND  + I2P
777     CONTINUE
778     FORMAT(6X,I5,1X,'-',1X,I5,10X,I3)
33      FORMAT(1612)
        STOP
        END
C
C
C       ************************************************
C       *                                              *
C       *     P I T C H     E X T R A C T I O N        *
C       *                                              *
C       ************************************************
C
C
C
        SUBROUTINE PITCH( IBUF, NP, ITMAX, ITMIN,  KBLK1, NUMP,NBUFF,IOPT2)
        DIMENSION A(200), IBUFF(512), IA(200), IBUF(512)
        COMMON /CLIPP/CLPP
C
C
        NUMP = NUMP + 1
        N1 = NBUFF/3
        N2 = N1+N1
        DO 5    I = 1,NBUFF
              IBUFF(I)=IBUF(I)
5       CONTINUE
        DO 10   I = 1, 200
```

```
                A(I) = Ø.Ø
               IA(I) = Ø
1Ø      CONTINUE
C
C------ CHECK IF CENTER CLIPPING IS ASKED FOR.
C
        IF(IOPT2 .LE. 2) GOTO 28Ø
C
C------ FIND OUT ABSOLUTE MAXIMUM OUT OF NBUFF SAMPLES IN THE BUFFER"IBUFF"
C
        IBIG1 = Ø
        IBIG2 = Ø
        DO 12Ø  I = 1,N1          ! FIND LARGEST SAMPLE IN 1ST OF 3 PARTS.
                ISPABS=ABS(IBUFF(I))
                IF(ISPABS .GT. IBIG1)IBIG1=ISPABS
12Ø     CONTINUE
C
C
        DO 121  I = N2, NBUFF     ! FIND LARGEST SAMPLE IN 3RD OF 3 PARTS.
                ISPABS=ABS(IBUFF(I))
                IF(ISPABS .GT. IBIG2)IBIG2=ISPABS
121     CONTINUE
        IB=IBIG1-IBIG2
        IF(IB .GE. Ø)IBIG=IBIG2   ! FIND MINIMUM OF TWO LARGE VALUES.
        IF(IB .LT. Ø)IBIG=IBIG1
C
C-----  ENTER THE CLIPPING LEVEL.
C
        CL = CLPP * FLOAT(IBIG)
C
C------ CHECK IF CENTER CLIPPING WITH THREE OR TWO VALUES IS REQUIRED.
C
        IF( IOPT2 .GT. 4) GOTO 155              ! YES, 2 OR 3 VALUE CLIPPING.
C
C------ CLIPP THE SPEECH WAVEFORM.
C
        CLM=-CL
        DO 14Ø  I=1,NBUFF               ! GENERATE CLIPPED SPEECH.
                XFLT=FLOAT(IBUFF(I))
                IF((XFLT .LE. CL) .AND. (XFLT .GT. CLM))GOTO 147
                IF( XFLT .GT. CL )IBUFF(I)=IBUFF(I)-IFIX(CL)
                IF( XFLT .LT. CLM)IBUFF(I)=IBUFF(I)+IFIX(CL)
                GOTO 14Ø
147             IBUFF(I)=Ø
14Ø     CONTINUE
        GOTO 28Ø
155     CONTINUE
        IF(IOPT2 .GT. 6)GOTO 36Ø                ! YES, 2-VALUE CLIPPING.
        CLM=-CL
        DO 16Ø  I=1,NBUFF               ! GENERATE 3-VALUE CLIPPED SP.
                XFLOT=FLOAT(IBUFF(I))
                IF((XFLOT .LE. CL) .AND. (XFLOT .GT. CLM))IBUFF(I)=Ø
                IF(XFLOT .GT. CL)IBUFF(I)=+1
                IF(XFLOT .LE. -CL)IBUFF(I)=-1
16Ø     CONTINUE
        IF(IOPT2 .GT. 5)GOTO 22Ø
C
C------ COMPUTE AUTOCORRELATION FUNCTIONS
C
        IBIGG=-1ØØØ
        DO 18Ø  IT=ITMIN,ITMAX          ! 3-VALUE C CLPPD AUTOCORR METHOD.
                ISUM=Ø
                DO 17Ø  J=1,KBLK1
                IF(((IBUFF(J+IT).LT.Ø).AND.(IBUFF(J).LT.Ø)) .OR. ((IBUFF(J+IT)
```

```
     1          .GT.Ø).AND.(IBUFF(J).GT.Ø)))ISUM=ISUM+1
                IF(((IBUFF(J+IT).LT.Ø).AND.(IBUFF(J).GT.Ø)) .OR. ((IBUFF(J+IT)
     1          .GT.Ø).AND.(IBUFF(J).LT.Ø)))ISUM=ISUM-1
17Ø             CONTINUE
          IA(IT)=ISUM
          IF(IBIGG .LT. IA(IT)) IBIGG=IA(IT)
18Ø       CONTINUE
          DO 19Ø  I=ITMIN,ITMAX
                  IF(IBIGG .EQ. IA(I))GOTO 2ØØ
19Ø       CONTINUE
2ØØ       NP=I
          WRITE(5,*)NP
          RETURN
C
C-----    CALCULATE AMDF FUNCTIONS
22Ø       CONTINUE
          ISMALL=4Ø96
          DO 23Ø  IT=ITMIN,ITMAX          ! 3-VALUE C CLPPD AUTOCORR METHOD.
                  ISUM=Ø
                  DO 24Ø  J=1,KBLK1
                          IF(((IBUFF(J+IT).EQ.Ø).AND.(IBUFF(J).NE.Ø)) .OR.
     1                    ((IBUFF(J+IT).NE.Ø).AND.(IBUFF(J).EQ.Ø)))ISUM=ISUM+1
                          IF(((IBUFF(J+IT).GT.Ø).AND.(IBUFF(J).LT.Ø)) .OR.
     1                    ((IBUFF(J+IT).LT.Ø).AND.(IBUFF(J).GT.Ø)))ISUM=ISUM+2
24Ø               CONTINUE
          IA(IT)=ISUM
          IF(ISMALL .GE. IA(IT))ISMALL=IA(IT)
23Ø       CONTINUE
          DO 25Ø  I=ITMIN,ITMAX
                  IF(ISMALL .EQ. IA(I))GOTO 26Ø
25Ø       CONTINUE
26Ø       NP=I
          WRITE(5,*)NP
          RETURN
28Ø       CONTINUE
          IF((IOPT2.EQ.2).OR.(IOPT2.EQ.4))GOTO 34Ø
          BIG=Ø.Ø
          DO 3ØØ  IT=ITMIN,ITMAX
                  SUM=Ø.Ø
                  DO 29Ø  J=1,KBLK1
                          SUM=SUM+FLOAT(IBUFF(J+IT))*FLOAT(IBUFF(J))
29Ø               CONTINUE
          A(IT)=SUM
          IF(BIG .LT. A(IT))BIG=A(IT)
3ØØ       CONTINUE
          DO 31Ø  I=ITMIN,ITMAX
                  IF(BIG.EQ.A(I))GOTO 32Ø
31Ø       CONTINUE
32Ø       NP=I
          WRITE(5,*)NP
          RETURN
34Ø       CONTINUE
          SMALL = 1.ØE+Ø9
          DO 6Ø   IT = ITMIN, ITMAX
                  SUM = Ø.Ø
                  DO 5Ø   J = 1, KBLK1
                          SUM = SUM + ABS(FLOAT(IBUFF(J+IT) - IBUFF(J)))
5Ø                CONTINUE
          A(IT) = SUM
          IF( SMALL .GE. A(IT)) SMALL = A(IT)
6Ø        CONTINUE
          DO 7Ø   I = ITMIN, ITMAX
                  IF(SMALL .EQ. A(I)) GOTO 8Ø
7Ø        CONTINUE
```

```
80      CONTINUE
        NP = I
        WRITE(5,*)NP
        RETURN
360     CONTINUE
C       CLIPP THE SPEECH TO TWO VALUES.
        CL=0.0
        DO 380  I = 1,NBUFF
                XFLOT=FLOAT(IBUFF(I))
                IF( XFLOT .LE. CL )IBUFF(I)=-1
                IF( XFLOT .GT. CL )IBUFF(I)=1
380     CONTINUE
        IF(IOPT2 .GT. 7)GOTO 480
        IBIGG = -1000
        DO 420  IT=ITMIN,ITMAX
                ISUM=0
                DO 400  J=1,KBLK1
                        IF(IBUFF(J+IT) .EQ. IBUFF(J))ISUM=ISUM+1
                        IF(IBUFF(J+IT) .NE. IBUFF(J))ISUM=ISUM-1
400             CONTINUE
        IA(IT)=ISUM
        IF(IBIGG .LT. IA(IT))IBIGG=IA(IT)
420     CONTINUE
        DO 440  I=ITMIN,ITMAX
                IF(IBIGG .EQ. IA(I))GOTO 460
440     CONTINUE
460     NP=I
        WRITE(5,*)NP
        RETURN
480     CONTINUE
        ISMALL=4096
        DO 500  IT=ITMIN,ITMAX
                ISUM=0
                DO 490  J=1,KBLK1
                        IF(IBUFF(J+IT) .NE. IBUFF(J))ISUM=ISUM+2
490             CONTINUE
        IA(IT)=ISUM
        IF(ISMALL .GE. IA(IT))ISMALL=IA(IT)
500     CONTINUE
        DO 520  I=ITMIN,ITMAX
                IF(ISMALL .EQ. IA(I))GOTO 540
520     CONTINUE
540     NP=I
        WRITE(5,*)NP
        RETURN
        END
C
C
C       ****************************************
C       *                                      *
C       *   W I N D O W    F U N C T I O N     *
C       *                                      *
C       ****************************************
C
C
C
        SUBROUTINE WINDOW(H,NP)
        DIMENSION H(400)
        DO 10   I = 1, NP
                H(I) = 1.0 - FLOAT(I-1)/FLOAT(NP-1)
10      CONTINUE
        RETURN
        END
CC
C
```

```
C       ********************************
C       *                              *
C       *    Q U A N T I Z E R         *
C       *                              *
C       ********************************
C
C       QUANTIZER IS VARIABLE LEVEL QUANTIZER.NUMBER OF LEVELS ARE KQ.
        SUBROUTINE QUANT(X,Y,I)
        INTEGER*2 F1,F2,F3,F4,STAT
        COMMON /QUAN/T(20),OUT(20),EXPN(20),SIZE,SMIN,NQ
        COMMON /RMS/RMS,NBL,IARG
        COMMON /ADDN/F1,F2,F3,F4,BT1,BT2,BT3
     1  ,JCNT,STAT(30),GAMA(4)
C
C
10      X1=ABS(X/SIZE)
        F=0.5
        IF(X.LT.0.)F=-.5
        I=1
        DO 20 K=1,NQ
        IF(JCNT .EQ. 1)TNEW=T(K)          ! OLD THRESHOLDS IF BUFFER SMALL.
        IF(JCNT .NE. 1)TNEW=GAMA(JCNT)+T(K)    ! NEW THRESHOLD
   20   IF(X1.GE.TNEW) I=2*K+.5+F
        J=(I+2)/2
        Y=2.*F*OUT(J)*SIZE
        SIZE=EXPN(J)*SIZE
        SIZE=AMAX1(SIZE,RMS*SMIN)
        RETURN
        END
C
C
C       ***********************************
C       *                                 *
C       *  I N I T I A L I Z A T I O N    *
C       *                                 *
C       ***********************************
C
C       PARAMETERS ARE DEFINED AND INITIALIZED.
C
        SUBROUTINE INSTRT
        INTEGER FILEN
        COMMON /QUAN/T(20),OUT(20),EXPN(20),SIZE,SMIN,NQ
        COMMON /PRED/G,N,RMSMIN,ALP,AINV,KQ,NSPSAM,A(12),VHAT(12),EV
     1  ,ISTAT1(40),EP,ALAD,SPERB1(200),SPERB2(200),V(12),SNRB(200)
     2  ,SNRQB(200)
        COMMON /RMS/RMS,NBL,IARG,ENGY1,ENGY2,ENGY3,ENGY4,SPER1,SPER2
        COMMON /INIT/JI
        COMMON /FN/FILEN(16)
C
C
        READ(3,40)FILEN          ! READ PARAMETERS FOR ARC SYSTEM.
   40   FORMAT(16A2)
        OPEN(UNIT=8,NAME=FILEN,TYPE='OLD')
        READ(8,*)AINV,ALP,ALAD,G,N,RMSMIN,SMIN
        WRITE(6,2)AINV,ALP,ALAD,G,N,RMSMIN,SMIN
    2   FORMAT(/6X,'AINV=',F5.2,2X,'ALP=',F5.2,2X,'ALAD=',F5.2,2X,'G='
     1  F5.3,2X,'N=',I2,2X,'RMSMIN=',F5.1,2X,'SMIN=',F5.2/)
        READ(8,*)KQ
        WRITE(6,3)KQ
    3   FORMAT(6X,'NUMBER OF QUANT LEVELS=',I2)
        NQ=KQ/2
        NQQQ=NQ+1
        READ(8,*)(EXPN(I),I=1,NQQQ)
        WRITE(6,4)(I,EXPN(I),I=1,NQQQ)
```

```
    4   FORMAT(6X,6('EXPN(',I2,')=',F6.2,2X))
        READ(8,*)(OUT(I),I=1,NQQQ)
        WRITE(6,5)(I,OUT(I),I=1,NQQQ)
    5   FORMAT(6X,6('OUT(',I2,')=',F6.2,2X))
   22   DO 3Ø I=1,NQ
   3Ø   T(I)=(OUT(I)+OUT(I+1))/2.
        SIZE=1ØØ.
        DO 118 I=1,12
        VHAT(I)=Ø.
        V(I)=Ø.
  118   A(I)=Ø.
        RMS=RMSMIN
        A(1)=AINV
        EV=Ø.
        EP=Ø.
        ENGY1=Ø.
        ENGY2=Ø.
        JI=Ø
        ENGY3=Ø.
        ENGY4=Ø.
        SPER1=Ø.
        SPER2=Ø.
        READ(3,*)NBL            ! BLOCK LENGTH TO CALCULATE SEGSNR.
        DO 621 I=1,KQ
  621   ISTAT1(I)=Ø
        RETURN
        END
C
C
C       ****************************************************
C       *                                                  *
C       *  A D A P T I V E   R E S I D U A L   C O D E R   *
C       *                                                  *
C       ****************************************************
C------  SUBROUTINE ARC  RETURNS QUANTIZER OUTPUT AND YHAT.
C
        SUBROUTINE ARC(Y,Q,YHAT)
        INTEGER Q
        COMMON /PRED/G,N,RMSMIN,ALP,AINV,KQ,NSPSAM,A(12),VHAT(12),
     1  EV,ISTAT1(4Ø),EP,ALAD,SPERB1(2ØØ),SPERB2(2ØØ),V(12),SNRB(2ØØ)
     2  ,SNRQB(2ØØ)
        COMMON /RMS/RMS,NBL,IARG,ENGY1,ENGY2,ENGY3,ENGY4,SPER1,SPER2
        COMMON /INIT/JI
C
C------ PREDICTION.
C
C
        PRE=Ø.
        PRE1=Ø.
        DO 12Ø I=1,N
                PRE1=PRE1+A(I)*V(I)
  12Ø   PRE=PRE+A(I)*VHAT(I)
        RMS=ALP*(RMS-RMSMIN)+(1.-ALP)*ABS(VHAT(1))+RMSMIN
        ERROR=Y-PRE
        ERROR1=Y-PRE1
        CALL QUANT(ERROR,EQ,IOUT)
        ISTAT1(IOUT)=ISTAT1(IOUT)+1
        Q=IOUT
        DO 125 I=1,N
        J=N+2-I
        V(J)=V(J-1)
  125   VHAT(J)=VHAT(J-1)
        VHAT(1)=PRE+EQ
        V(1)=Y
```

```
      YHAT=VHAT(1)
C
C
C------ ADAPTATION.
C
      ERR=G*EQ/RMS**2
      A(1)=A(1)+AINV*(1./ALAD-1.)
      DO 13Ø I=1,N
  13Ø A(I)=A(I)*ALAD+ERR*VHAT(I+1)
C
C
C------ UPDATE PAST.
C
      IRM=MOD(IARG,NBL)
      ENGY1=ENGY1+Y**2
      ENGY2=ENGY2+(ERROR-EQ)**2
      ENGY3=ENGY3+ERROR**2
      ENGY4=ENGY4+ERROR1**2
      IF(IRM .NE. Ø)GOTO 133
      JI=JI+1
      EV=EV+ENGY1
      EP=EP+ENGY2
      IF(ENGY3 .NE. Ø.)SPERB1(JI)=1Ø.*ALOG1Ø(ENGY1/ENGY3)
      IF(ENGY4 .NE. Ø.)SPERB2(JI)=1Ø.*ALOG1Ø(ENGY1/ENGY4)
      IF(ENGY2 .NE. Ø.)SNRB(JI)=1Ø.*ALOG1Ø(ENGY1/ENGY2)
      IF(ENGY2 .NE. Ø.)SNRQB(JI)=1Ø.*ALOG1Ø(ENGY3/ENGY2)
      SPER1=SPER1+ENGY3
      SPER2=SPER2+ENGY4
      ENGY1=Ø.
      ENGY2=Ø.
      ENGY3=Ø.
      ENGY4=Ø.
133   CONTINUE
      RETURN
      END
C
C
C     ******************************************
C     *                                        *
C     *    STATISTICS   AND   RESULTS          *
C     *                                        *
C     ******************************************
C------ SUBROUTINE INEND WRITES ALL THE RESULTS,SUCH AS SNR,H
C------ AND STATISTICS.
C
      SUBROUTINE INEND
      REAL PROB(3Ø)
      INTEGER*2 STAT,F1,F2,F3,F4
      COMMON /PRED/G,N,RMSMIN,ALP,AINV,KQ,NSPSAM,A(12),
     1 VHAT(12),EV,ISTAT1(4Ø),EP,ALAD,SPERB1(2ØØ),SPERB2(2ØØ),V(12)
     2 ,SNRB(2ØØ),SNRQB(2ØØ)
      COMMON /RMS/RMS,NBL,IARG,ENGY1,ENGY2,ENGY3,ENGY4,SPER1,SPER2
      COMMON /ADDN/F1,F2,F3,F4,BT1,BT2,BT3
     1 ,JCNT,STAT(3Ø),GAMA(4)
C
C
      SNR=1Ø.*ALOG1Ø(EV/EP)
      SPER=1Ø.*ALOG1Ø(EV/SPER1)
      SPERI=1Ø.*ALOG1Ø(EV/SPER2)
      SNRQ=1Ø.*ALOG1Ø(SPER1/EP)
      ISUM=Ø
      DO 3ØØ II=1,KQ
3ØØ   ISUM=ISUM+ISTAT1(II)
      ARG1=ISUM
```

```
      SUM=ALOG(ARG1)
      DO 500 I=1,KQ
      ARG=ISTAT1(I)+0.001
  500 SUM=SUM-(ARG*ALOG(ARG))/ARG1
      BITS=SUM/ALOG(2.)
      WRITE(6,402)SNR,BITS,(ISTAT1(I),I=1,KQ)
  402 FORMAT(6X,'SNR INLOOP=',F7.3,3X,'H=',F4.2,3X,'OP',
     1  10I5,3(/22X,10I5)//)
      WRITE(5,402)SNR,BITS,(ISTAT1(I),I=1,KQ)
      WRITE(6,404)
404   FORMAT(///6X,' SAMPLE NUMBER ',6X,' SNR ',6X,' SPER ',6X,
     1  ' SPERI ',6X,' SNRQ '//)
C
      NB=ISUM/NBL
C
      DO 408  I=1,NB
              IS=(I-1)*NBL+1
              IE=IS+NBL-1
              WRITE(6,412)IS,IE,SNRB(I),SPERB1(I),SPERB2(I),SNRQB(I)
408   CONTINUE
412   FORMAT(6X,I5,'-',I5,6X,F7.2,4X,F7.2,4X,F7.2,4X,F7.2)
      WRITE(6,416)SPER,SPERI,SNRQ
416   FORMAT(//6X,' PREDICTOR PERFORMANCE =',F8.2/6X,
     1  ' PREDICTOR IDEAL PERFORMANCE= ',F8.2/6X,' SIGNAL TO NOISE
     2  RATIO=',F8.2)
C
C
      NQUA=0
      DO 418  I=1,16
418   NQUA=NQUA+STAT(I)
      DO 420  I=1,16
              PROB(I)=FLOAT(STAT(I))/FLOAT(NQUA)
420   CONTINUE
      WRITE(6,422)
422   FORMAT(//,6X,'LEVEL NUMBER',6X,'PROBABILITY',6X,' FREQUENCY',/)
424   FORMAT(9X,I2,12X,F7.4,10X,I5)
      DO 426  I=1,16
      WRITE(6,424)I,PROB(I),STAT(I)
426   CONTINUE
      RETURN
      END
C
C     ******************************************
C     *                                        *
C     *         BUFFER    CONTROL              *
C     *                                        *
C     ******************************************
      SUBROUTINE BUFCTL(BUFCNT,J,AVGBIT)
      INTEGER*2  STAT,F1,F2,F3,F4
      COMMON /ADDN/F1,F2,F3,F4,BT1,BT2,BT3
     1  ,JCNT,STAT(30),GAMA(4)
C
C
              IF( F1 .EQ. 1)GOTO 50
              IF( F2 .EQ. 1)GOTO 60
              IF( F3 .EQ. 1)GOTO 70
              IF( J .LE. 3) GOTO 80
              STAT(J)=STAT(J)+1
              BUFCNT=BUFCNT+4.-AVGBIT
              F4=0
              GOTO 100
50            CONTINUE
              IF(J .EQ. 1)GOTO 53
              IF(J .EQ. 2)GOTO 153
```

```
                IF(J .EQ. 3)GOTO 155
                STAT(J)=STAT(J)+1
                STAT(1)=STAT(1)+1
                F1=Ø
                BUFCNT=BUFCNT+4.-AVGBIT
                BUFCNT=BUFCNT+4.-AVGBIT
                F4=1
                GOTO 1ØØ
53              CONTINUE
                STAT(12)=STAT(12)+1
                BUFCNT=BUFCNT+4.-AVGBIT-AVGBIT
                F4=Ø
                F1=Ø
                GOTO 1ØØ
153             F2=1
                STAT(1)=STAT(1)+1
                F1=Ø
                BUFCNT=BUFCNT+4.-AVGBIT
                GOTO 1ØØ
155             F3=1
                STAT(1)=STAT(1)+1
                F1=Ø
                BUFCNT=BUFCNT+4.-AVGBIT
                GOTO 1ØØ
6Ø              CONTINUE
                IF( J .EQ. 1)GOTO 63
                IF( J .EQ. 2)GOTO 65
                IF( J .EQ. 3)GOTO 67
                STAT(2)=STAT(2)+1
                BUFCNT=BUFCNT+4.-AVGBIT
                STAT(J)=STAT(J)+1
                BUFCNT=BUFCNT+4.-AVGBIT
                F4=1
                F2=Ø
                GOTO 1ØØ
63              STAT(13)=STAT(13)+1
                BUFCNT=BUFCNT+4.-AVGBIT-AVGBIT
                F2=Ø
                F4=Ø
                GOTO 1ØØ
65              STAT(14)=STAT(14)+1
                BUFCNT=BUFCNT+4.-AVGBIT-AVGBIT
                F4=Ø
                F2=Ø
                GOTO 1ØØ
67              STAT(2)=STAT(2)+1
                BUFCNT=BUFCNT+4.-AVGBIT
                F3=1
                F2=Ø
                F4=Ø
                GOTO 1ØØ
7Ø              CONTINUE
                IF( J .EQ. 1)GOTO 72
                IF( J .EQ. 2)GOTO 74
                IF( J .EQ. 3)GOTO 76
                STAT(3)=STAT(3)+1
                BUFCNT=BUFCNT+4.-AVGBIT
                STAT(J)=STAT(J)+1
                BUFCNT=BUFCNT+4.-AVGBIT
                F3=Ø
                F4=1
                GOTO 1ØØ
72              STAT(15)=STAT(15)+1
                BUFCNT=BUFCNT+4.-AVGBIT-AVGBIT
```

```
                    F3=Ø
                    F4=Ø
                    GOTO 1ØØ
74                  STAT(3)=STAT(3)+1
                    BUFCNT=BUFCNT+4.-AVGBIT
                    F2=1
                    F4=Ø
                    F3=Ø
                    GOTO 1ØØ
76                  STAT(16)=STAT(16)+1
                    BUFCNT=BUFCNT+4.-AVGBIT-AVGBIT
                    F4=Ø
                    F3=Ø
                    GOTO 1ØØ
8Ø                  CONTINUE
                    IF( J .EQ. 1)F1=1
                    IF( J .EQ. 2)F2=1
                    IF( J .EQ. 3)F3=1
1ØØ         CONTINUE
            IF(BUFCNT .GT. BT1)JCNT=2
            IF(BUFCNT .GT. BT2)JCNT=3
            IF(BUFCNT .GT. BT3)JCNT=4
            IF(BUFCNT .LT. BT1)JCNT=1
            IF(BUFCNT .LT. Ø.)BUFCNT=Ø.
            RETURN
            END
```

```
C       ******************************************
C       *                                        *
C       *        ENTROPY   CODING                *
C       *                                        *
C       ******************************************
C
C
C       PROGRAM NAME:          ENCO.FTN
C
C       DATE:                  JUNE 3Ø,1981
C
C
C
C       THIS PROGRAM READS THE QUANTIZER LEVELS, SORTS THEM OUT ACCORDING TO
C       RUN LENGTH AND PUTS APPROPRIATE CODE WORDS IN OUTPUT FILE.
C
        INTEGER*2 BIN(256),BOUT(512),TBU(256),HD(4Ø),BOUT1(256)
        DIMENSION IVECT(11)
        EQUIVALENCE (BOUT1(1),BOUT(1))
        DATA IVECT/Ø,1,4,2,5,3,6,11,7,15,14/
        KBIT=Ø
        LAST=Ø
        NEWBLO=Ø
        TYPE *,' TYPE THRESHOLD TO SWITCH CODE TO PREVENT BUFFER UNDERFLOW'
        TYPE *,' TYPICAL VALUE: 1ØØ'
        ACCEPT *,ITH
        IBCNT=Ø
C
C
        OPEN(UNIT=1,TYPE='OLD',NAME='QUANT.DAT')
        OPEN(UNIT=2,TYPE='NEW',NAME='CODE.DAT',CARRIAGECONTROL='LIST')
        OPEN(UNIT=3,TYPE='NEW',NAME='TBUF.DAT',CARRIAGECONTROL='LIST')
C
C
        TYPE *,' TYPE HAMMING CODE:INFO BITS,CHECK BITS'
        ACCEPT *,INFOB,NPARI
C
C
        READ(1,7)NSENT,IRATE,NSAMP,IUPPR,ILOWR,NTERMS,HD
7       FORMAT(6I5,1ØX,4ØA1)
        TYPE *,' TYPE THE HEADER FOR BUFFER FILE:'
        ACCEPT 8,HD
8       FORMAT(4ØA1)
        WRITE(3,7)NSENT,IRATE,NSAMP,IUPPR,ILOWR,NTERMS,HD
        TYPE *,' TYPE HEADER FOR ENCODER OUTPUT FILE:'
        ACCEPT 8,HD
       WRITE(2,7)NSENT,IRATE,NSAMP,IUPPR,ILOWR,NTERMS,HD
C
C
        NBLO = NSAMP/256
        J=Ø
        DO 1Ø    IB=1,NBLO                  ! BLOCK LOOP
                 READ(1,1ØØ,END=333)BIN
1ØØ              FORMAT(16I2)
                 IP=1
                 I=Ø
13               I=I+IP
                        J=J+1
                        M=BIN(I)
                        IF(I .EQ. 256 .OR. KBIT .LE. ITH)GOTO 6Ø
                        M1=BIN(I+1)
                        IF(M.GT.3  .OR. M1 .GT. 3)GOTO 6Ø
                        IF(M .EQ. 1  .AND. M1 .GT. 3)GOTO 6Ø
                        IF(M .EQ. 1  .AND. M1 .EQ. 2)GOTO 6Ø
```

```
                        IF(M .EQ. 1   .AND. M1 .EQ. 3)GOTO 6Ø
                        IF(M .EQ. 2   .AND. M1 .EQ. 3)GOTO 6Ø
                        IF(M .EQ. 3   .AND. M1 .EQ. 2)GOTO 6Ø
                        IF((M .EQ. 1) .AND. (M1.EQ.1))KCOD=8
                        IF((M .EQ. 2) .AND.(M1.EQ.1))KCOD=1Ø     ! 2,1
                        IF((M .EQ. 2) .AND.(M1.EQ.2))KCOD=13     ! 2,2
                        IF((M .EQ. 3) .AND.(M1.EQ.1))KCOD=9      ! 3,1
                        IF((M .EQ. 3) .AND.(M1.EQ.3))KCOD=12     ! 3,3
                        IP=2
                        BOUT(J)=KCOD
                        KBIT=KBIT-2
                        DO 78    ILOOP=1,4           ! ADD "NPARI" BITS
                                 IBCNT=IBCNT+1       ! TO BUFFER EVERY "INFOB"
                                 IF(MOD(IBCNT,INFOB) .NE. Ø)GOTO 78
                                 KBIT=KBIT+NPARI
                                 IBCNT=Ø
78                      CONTINUE
C
C
        IF(KBIT.GT.1Ø24)KBIT=1Ø24
        IF(KBIT.LT.Ø)KBIT=Ø
        IF(I.NE.1)GOTO 8Ø
        TBU(I)=LAST
        GOTO 81
 8Ø     TBU(I)=TBU(I-1)
 81     TBU(I+1)=KBIT
                        GOTO 2Ø
6Ø                      IP=1
                        BOUT(J)=IVECT(M)
                        KBIT=KBIT+1
C
                        DO 62    ILOOP=1,4
                                 IBCNT=IBCNT+1
                                 IF((MOD(IBCNT,INFOB)) .NE. Ø)GOTO 62
                                 KBIT=KBIT+NPARI
                                 IBCNT=Ø
62                      CONTINUE
C
C
        IF(KBIT.GT.1Ø24)KBIT=1Ø24
        IF(KBIT.LT.Ø)KBIT=Ø
        TBU(I)=KBIT
2Ø              CONTINUE
C
C
C       WRITE(5,3ØØ)I,BIN(I),(I+1),BIN(I+1),J,BOUT(J),KCOD,TBU(J)
C3ØØ    FORMAT(2X,'BIN(',I3,')=',I3,2X,'BIN(',I3,')=',I3,2X,'BOUT(',I3,')=',
C     1 I5,2X,'KCOD=',I2,2X,'KBIT=',I4)
C
C
                IF(I.EQ.256 .OR. (I.EQ.255 .AND. IP .EQ. 2))GOTO 7Ø
                GOTO 13
C
C
7Ø              WRITE(3,2ØØ)TBU
2ØØ             FORMAT(16I5)
                LAST=TBU(256)
                IF(J .LT. 256)GOTO 1Ø
        WRITE(2,15Ø)BOUT1
15Ø     FORMAT(16I2)
                NEWBLO=NEWBLO+1
                KK=J-256
                IF(KK .EQ. Ø)GOTO 93
                DO 91    II=1,KK
```

```
91              BOUT(II)=BOUT(256+II)
93              J=KK
10      CONTINUE
C
C
        IF(J .LT. 256) GOTO 410
        IF(J .GT. 256) GOTO 550
        GOTO 333
C
C
410     LL=256-J
        DO 420  L=1,LL
420     BOUT(J+L)=IVECT(1)
430     WRITE(2,150)BOUT1
        NEWBLO=NEWBLO+1
        GOTO 333
C
C
550     DO 560  I=1,256
560     BOUT1(I)=IVECT(1)
        KK=J-256
        DO 570  II=1,KK
570     BOUT(II)=BOUT(256+II)
        GOTO 430
C
C
333     CLOSE (UNIT=1)
        CLOSE (UNIT=2)
        CLOSE (UNIT=3)
C
C
        OPEN(UNIT=1,TYPE='NEW',NAME='NEWBLO.DAT')
        WRITE(1,666)NEWBLO
666     FORMAT(I3)
        CLOSE(UNIT=1)
C
C
        STOP
        END
```

```
C     ************************************************
C     *                                              *
C     *      C H A N N E L   E R R O R               *
C     *                                              *
C     *      WITH PARITY CHECK AND/OR                *
C     *                                              *
C     *      DOUBLE ERROR CORRECTION                 *
C     *                                              *
C     ************************************************
C
C
C     PROGRAM NAME:   CHER2631.FTN
C
C
C
C     THIS PROGRAM READS  CODE.DAT  FILE  AND INTRODUCES RANDOM ERRORS
C     IN THE BIT STREAM ACCORDING TO BIT ERROR RATE (BER) IN PERCENT.
C     THEN IT WRITES A NEW FILE WITH CODEWORD AFFECTED BY CHANNEL ERRORS.
C     AS OPTION, IT PERFORMS PARITY CHECK CORRECTION AND/OR DOUBLE
C     ERROR CORRECTION.
C     THE FRAME LOOP OF 32 BITS COULD BE ASSOCIATED TO A (31,26)
C     HAMMING CODE WITH  6*4 + 1 SYNC. + 1 DOUBLE ERR.DETECT. =26
C
C
C
C
C
      INTEGER*2 BIN(256),BOUT(256),HD(40),FLAG,ISAM(30),DOUB
      TYPE *,' ENTER  BER(%)'
      ACCEPT *,BER
      TYPE *,' PARITY CHECK = 1 ; NO PAR. CHECK = 0'
      ACCEPT *,FLAG
      TYPE *,' DOUBLE ERR. CORRECT. = 1 ; NO ERR. CORRECT. = 0'
      ACCEPT *,DOUB
C
      IDO=0
      ICS=0
      ICD=0
      ICT=0
      ICOR=0
      IF(BER .LT. 1.E-07)BER=0.0
      EXT = BER/100.
      K1  = 773
      K2  = 119
      CONT=0.
      IUNO=1
C
C
      OPEN(UNIT=1,TYPE='OLD',NAME='CODE.DAT')
      OPEN(UNIT=2,TYPE='NEW',NAME='ERCO.DAT',CARRIAGECONTROL='LIST')
      OPEN(UNIT=3,TYPE='OLD',NAME='NEWBLO.DAT')
      READ(3,5)NBLO
5     FORMAT(I3)
      READ(1,9)NSENT,IRATE,NBBS,IUPPR,ILOWR,NTERMS,HD
9     FORMAT(6I5,10X,40A1)
      TYPE *,' TYPE HEADER FOR CHERR OUTPUT FILE:'
      ACCEPT 11,HD
11    FORMAT(40A1)
      WRITE(2,9)NSENT,IRATE,NBLO,IUPPR,ILOWR,NTERMS,HD
C
C
      DO 13   JJ=1,10000
13    CALL RANDU(K1,K2,X)
```

```
      DO 1Ø    IB=1,NBLO                    ! BLOCK LOOP
               READ(1,1ØØ,END=345)BIN
1ØØ            FORMAT(16I2)
C
C
               DO 2Ø    IR=1,32             ! SAMPLE LOOP
               IKK=Ø
               DO 22 JG=1,3Ø
22             ISAM(JG)=Ø
               L=Ø
C
               DO 25 IZ=1,8              ! FRAME LOOP 8*4
               I=IZ+8*(IR-1)
                        M=BIN(I)
                        DO 3Ø    II=1,4
                                 CALL RANDU(K1,K2,X)
                                 IF(X .GT. EXT) GOTO 3Ø
                                 J = (2. ** (II-1) + Ø.5)
                                 M = IEOR(M,J)
                                 L=L+1
                                 ISAM(L)=I
                                 IKK=IKK+1
                                 CONT=CONT+1.
3Ø                       CONTINUE
                         BOUT(I)=M
25             CONTINUE
      IF(IKK.NE.1)GOTO 65
      ICS=ICS+1
      IF(FLAG.EQ.Ø)GOTO 2Ø
      IX=ISAM(1)
      BOUT(IX)=BIN(IX)
      ICOR=ICOR+1
      GOTO 2Ø
C
65    IF(IKK.EQ.Ø)GOTO 2Ø
      IF(IKK.EQ.2 .AND. DOUB.EQ.1)GOTO 75
      GOTO 8Ø
75    IDO=IDO+1
      IX=ISAM(1)
      BOUT(IX)=BIN(IX)
      IX=ISAM(2)
      BOUT(IX)=BIN(IX)
8Ø    IKK=IAND(IKK,IUNO)
      IF(IKK.EQ.Ø)ICD=ICD+1
      IF(IKK.EQ.1)ICT=ICT+1
      IF(IKK.EQ.1.AND.FLAG.EQ.1)GOTO 99
               GOTO 2Ø
C
C
99             OK1=K1
               OK2=K2
               DO 7ØØ IPLUS=1,5Ø      ! EXTRA LOOPS
               DO 71Ø IZ=1,8          ! FRAME LOOP
               I=IZ+8*(IR-1)
C
                        DO 72Ø KF=1,3Ø
                        IPR=ISAM(KF)
                        IF(IPR.EQ.I)GOTO 71Ø
72Ø                     CONTINUE
C
               M=BIN(I)
               DO 73Ø II=1,4
               CALL RANDU(K1,K2,X)
               IF(X.GT.EXT)GOTO 73Ø
```

```
                J=2.**(II-1) + 0.5
                M=IEOR(M,J)
                CONT=CONT+1.
                GOTO 780
730             CONTINUE
710             CONTINUE
700             CONTINUE
C
780     BOUT(I)=M       !  INSERT ONE ADDITIONAL ERROR : PARITY CHECK FAILS
                K1=OK1
                K2=OK2
C
C
20              CONTINUE
C
C
C
                WRITE(2,150)BOUT
150             FORMAT(16I2)
10      CONTINUE
345     CONTINUE
        TYPE *,' TOTAL # OF ERRORS=',CONT
        TYPE *,' TOTAL # OF SINGLE ERRORS CORRECTED= ',ICOR
        TYPE *,' TOTAL # OF DOUBLE ERRORS CORRECTED= ',IDO
        TYPE *,' TOTAL # OF SINGLE ERR./FRAME= ',ICS
        TYPE *,' TOTAL # OF EVEN ERR./FRAME= ',ICD
        TYPE *,' TOTAL # OF ODD ERR./FRAME= ',ICT
        RESI=CONT-ICOR-2*IDO
        TYPE *,' # OF RESIDUAL ERRORS= ',RESI
        IF(CONT .NE. 0.)BERNEW=(RESI/CONT)*BER
        IF(CONT .NE. 0.)TYPE *,' NEW BER AFTER ERR CORRECTION = ',BERNEW
        CLOSE(UNIT=1)
        CLOSE(UNIT=2)
        CLOSE(UNIT=3)
        STOP
        END
```

```
C       ************************************************
C       *                                              *
C       *      C H A N N E L   E R R O R               *
C       *                                              *
C       *      WITH PARITY CHECK AND/OR                *
C       *                                              *
C       *      DOUBLE ERROR CORRECTION                 *
C       *                                              *
C       ************************************************
C
C
C       PROGRAM NAME:   CHER5763.FTN
C
C
C
C
C       THIS PROGRAM READS  CODE.DAT  FILE  AND INTRODUCES RANDOM ERRORS
C       IN THE BIT STREAM ACCORDING TO BIT ERROR RATE (BER) IN PERCENT.
C       THEN IT WRITES A NEW FILE WITH CODEWORD AFFECTED BY CHANNEL ERRORS.
C       AS OPTION, IT PERFORMS PARITY CHECK CORRECTION AND/OR DOUBLE
C       ERROR CORRECTION.
C
C
C
C
        INTEGER*2 BIN(256),BOUT(256),HD(40),FLAG,ISAM(30),DOUB
        TYPE *,' ENTER  BER(%)'
        ACCEPT *,BER
        TYPE *,' PARITY CHECK = 1 ; NO PAR. CHECK = 0'
        ACCEPT *,FLAG
        TYPE *,' DOUBLE ERR. CORRECT. = 1 ; NO ERR. CORRECT. = 0'
        ACCEPT *,DOUB
C
        IDO=0
        ICS=0
        ICD=0
        ICT=0
        ICOR=0
        IF(BER .LT. 1.E-07)BER=0.0
        EXT = BER/100.
        K1  = 773
        K2  = 119
        CONT=0.
        IUNO=1
C
C
        OPEN(UNIT=1,TYPE='OLD',NAME='CODE.DAT')
        OPEN(UNIT=2,TYPE='NEW',NAME='ERCO.DAT',CARRIAGECONTROL='LIST')
        OPEN(UNIT=3,TYPE='OLD',NAME='NEWBLO.DAT')
        READ(3,5)NBLO
5       FORMAT(I3)
        READ(1,9)NSENT,IRATE,NBBS,IUPPR,ILOWR,NTERMS,HD
9       FORMAT(6I5,10X,40A1)
        TYPE *,' TYPE HEADER FOR CHERR OUTPUT FILE:'
        ACCEPT 11,HD
11      FORMAT(40A1)
        WRITE(2,9)NSENT,IRATE,NBLO,IUPPR,ILOWR,NTERMS,HD
C
C
        DO 13   JJ=1,10000
13      CALL RANDU(K1,K2,X)
        DO 10   IB=1,NBLO                       ! BLOCK LOOP
                READ(1,100,END=345)BIN
100             FORMAT(16I2)
```

```
C
C
                DO 2Ø    IR=1,16                ! SAMPLE LOOP
                IKK=Ø
                DO 22 JG=1,3Ø
22              ISAM(JG)=Ø
                L=Ø
C
                DO 25 IZ=1,16                  ! FRAME LOOP 16*4
                I=IZ+16*(IR-1)
                        M=BIN(I)
                        DO 3Ø    II=1,4
                                 CALL RANDU(K1,K2,X)
                                 IF(X .GT. EXT) GOTO 3Ø
                                 J = (2. ** (II-1) + Ø.5)
                                 M = IEOR(M,J)
                                 L=L+1
                                 ISAM(L)=I
                                 IKK=IKK+1
                                 CONT=CONT+1.
3Ø                      CONTINUE
                        BOUT(I)=M
25              CONTINUE
        IF(IKK.NE.1)GOTO 65
        ICS=ICS+1
        IF(FLAG.EQ.Ø)GOTO 2Ø
        IX=ISAM(1)
        BOUT(IX)=BIN(IX)
        ICOR=ICOR+1
        GOTO 2Ø
C
65      IF(IKK.EQ.Ø)GOTO 2Ø
        IF(IKK.EQ.2 .AND. DOUB.EQ.1)GOTO 75
        GOTO 8Ø
75      IDO=IDO+1
        IX=ISAM(1)
        BOUT(IX)=BIN(IX)
        IX=ISAM(2)
        BOUT(IX)=BIN(IX)
8Ø      IKK=IAND(IKK,IUNO)
        IF(IKK.EQ.Ø)ICD=ICD+1
        IF(IKK.EQ.1)ICT=ICT+1
        IF(IKK.EQ.1.AND.FLAG.EQ.1)GOTO 99
                GOTO 2Ø
C
C
99              OK1=K1
                OK2=K2
                DO 7ØØ IPLUS=1,5Ø       ! EXTRA LOOPS
                DO 71Ø IZ=1,16          ! FRAME LOOP
                I=IZ+16*(IR-1)
C
                        DO 72Ø KF=1,3Ø
                        IPR=ISAM(KF)
                        IF(IPR.EQ.I)GOTO 71Ø
72Ø                     CONTINUE
C
                M=BIN(I)
                DO 73Ø II=1,4
                CALL RANDU(K1,K2,X)
                IF(X.GT.EXT)GOTO 73Ø
                J=2.**(II-1) + Ø.5
                M=IEOR(M,J)
                CONT=CONT+1.
```

```
                GOTO 78Ø
73Ø             CONTINUE
71Ø             CONTINUE
7ØØ             CONTINUE
C
78Ø     BOUT(I)=M       !  INSERT ONE ADDITIONAL ERROR : PARITY CHECK FAILS
                K1=OK1
                K2=OK2
C
C
2Ø              CONTINUE
C
C
C
                WRITE(2,15Ø)BOUT
15Ø             FORMAT(16I2)
1Ø      CONTINUE
345     CONTINUE
        TYPE *,' TOTAL # OF ERRORS=',CONT
        TYPE *,' TOTAL # OF SINGLE ERRORS CORRECTED= ',ICOR
        TYPE *,' TOTAL # OF DOUBLE ERRORS CORRECTED= ',IDO
        TYPE *,' TOTAL # OF SINGLE ERR./FRAME= ',ICS
        TYPE *,' TOTAL # OF EVEN ERR./FRAME= ',ICD
        TYPE *,' TOTAL # OF ODD ERR./FRAME= ',ICT
        RESI=CONT-ICOR-2*IDO
        TYPE *,' # OF RESIDUAL ERRORS= ',RESI
        IF(CONT .NE. Ø.)BERNEW=(RESI/CONT)*BER
        IF(CONT .NE. Ø.)TYPE *,' NEW BER AFTER CORRECTION= ',BERNEW
        CLOSE(UNIT=1)
        CLOSE(UNIT=2)
        CLOSE(UNIT=3)
        STOP
        END
```

```
C       ******************************************
C       *                                        *
C       *          D E C O D E R                 *
C       *                                        *
C       ******************************************
C
C
C       PROGRAM NAME:      DECO3.FTN
C
C
C
C       THIS PROGRAM READS CODEWORD WITH CHANNEL ERRORS AND DECODES THEM
C       AND WRITES CORRESPONDING QUANTIZER LEVELS IN A NEW FILE. IT ALSO
C       COMPUTES RECEIVER BUFFER OCCUPANCY.
C
        REAL SAMP(1024),LAST
        INTEGER*2 BIN(256),BOUT(1024),ISAMP(256),FNAME(16),KAR(2),BOUT1(256)
        INTEGER*2 HD(40),HD1(40)
        EQUIVALENCE (BOUT(1),BOUT1(1))
C
        TYPE *,' ENTER THE CODEWORD FILENAME FOR DECODER:'
        ACCEPT 4,FNAME
4       FORMAT(16A2)
C
        OPEN(UNIT=1,TYPE='OLD',NAME=FNAME)
        OPEN(UNIT=2,TYPE='NEW',NAME='DECO.DAT',CARRIAGECONTROL='LIST')
        OPEN(UNIT=3,TYPE='NEW',NAME='RBUF.DAT',CARRIAGECONTROL='LIST')
        OPEN(UNIT=4,TYPE='OLD',NAME='NEWBLO.DAT')
        READ(4,5)NBLO
5       FORMAT(I3)
C
C
       READ(1,9)NSENT,IRATE,NBOLD,IUPPR,ILOWR,NTERMS,HD
        NSAMP=NBLO*256
9      FORMAT(6I5,10X,40A1)
       TYPE *,' TYPE HEADER FOR DECODER OUTPUT FILE:'
        ACCEPT 11,HD
11      FORMAT(40A1)
       WRITE(2,9)NSENT,IRATE,NSAMP,IUPPR,ILOWR,NTERMS,HD
       TYPE *,' TYPE HEADER FOR RECEIVER BUFFER OUTPUT FILE:'
        ACCEPT 11,HD1
       WRITE(3,9)NSENT,IRATE,NSAMP,IUPPR,ILOWR,NTERMS,HD1
C
C
        TYPE *,' INITIALIZE THE RECEIVER SAMPLE BUFFER:'
        ACCEPT *,WORD
C       LAST=300.      !RECEIVER BUFFER INITIALIZ.
C       WORD=300.      !  "" ""    "" ""   "" ""
        LAST=WORD
        IND=1
        IEND=0
        NEWB=0
C
C
        DO 10    IB=1,NBLO
                 READ(1,100,END=345)BIN
100              FORMAT(16I2)
                 DO 20    I=1,256
                          L=BIN(I)
                          IF(L .EQ. 8)GOTO 35
                          IF(L .EQ. 10)GOTO 40
                          IF(L .EQ. 13)GOTO 45
                          IF(L .EQ. 9)GOTO 50
                          IF(L .EQ. 12)GOTO 53
```

```
                    IF(L .EQ. 0)BOUT(IND)=1
                    IF(L .EQ. 1)BOUT(IND)=2
                    IF(L .EQ. 2)BOUT(IND)=4
                    IF(L .EQ. 3)BOUT(IND)=6
                    IF(L .EQ. 4)BOUT(IND)=3
                    IF(L .EQ. 5)BOUT(IND)=5
                    IF(L .EQ. 6)BOUT(IND)=7
                    IF(L .EQ. 7)BOUT(IND)=9
                    IF(L .EQ. 11)BOUT(IND)=8
                    IF(L .EQ. 14)BOUT(IND)=11
                    IF(L .EQ. 15)BOUT(IND)=10
                    WORD=WORD+0.25-0.3333  !1SAMPLE/4BITS=0.25
                    IF(WORD .GT. 350.)WORD=350.
                    IF(WORD .LT. 0.)WORD=0.
                    SAMP(IND)=WORD
                    IND=IND+1
                    GOTO 20
35                  KAR(1)=1
                    KAR(2)=1
                    GOTO 60
40                  KAR(1)=2
                    KAR(2)=1
                    GOTO 60
45                  KAR(1)=2
                    KAR(2)=2
                    GOTO 60
50                  KAR(1)=3
                    KAR(2)=1
                    GOTO 60
53                  KAR(1)=3
                    KAR(2)=3
C
C
C
C
60                  BOUT(IND)=KAR(1)
                    BOUT(IND+1)=KAR(2)
                    WORD=WORD+0.5-0.3333  !2SAMPLES/4BITS=0.5
                    IF(WORD .GT. 350.)WORD=350.
                    IF(WORD .LT. 0.)WORD=0.
                    IF(IND .NE. 1)GOTO 80
                    SAMP(IND)=LAST
                    GOTO 81
80                  SAMP(IND)=SAMP(IND-1)
81                  SAMP(IND+1)=WORD
                    IND=IND+2
C
C
20              CONTINUE    ! SAMPLE LOOP
C
C
149             WRITE(2,150)BOUT1
150             FORMAT(16I2)
                NEWB=NEWB+1
C
                DO 55   I=1,256
55              ISAMP(I)=SAMP(I)+0.5
                WRITE(3,151)ISAMP
151             FORMAT(16I5)
C
                IF(IND .LE. 257) GOTO 75
                IF(IB.EQ.NBLO)IEND=1
        IF(IND .GT. 1024)STOP 'BUFFER TOO SHORT'
                KK=IND-257
```

```
                DO 7Ø   I=I,KK
                BOUT(I)=BOUT(256+I)
7Ø              SAMP(I)=SAMP(256+I)
                IND=KK+1
                IF(IND .GT. 257)GOTO 149
                GOTO 1Ø
75              IND=1
                LAST=SAMP(256)
C
C
1Ø      CONTINUE        !  BLOCK LOOP
C
C
        IF(IND .LT. 257 .AND. IEND .EQ. Ø)GOTO 345
C
        WRITE(5,987)IND
987     FORMAT(1H ,'# SAMPLES IN THE TAIL = ',I6)
        IF(IND .GE. 257)GOTO 43Ø
C
                DO 3Ø1  I=IND,256
                        BOUT(I)=I       ! FILL WITH SILENCE
                        WORD=WORD+Ø.25-Ø.3333
                        IF(WORD .GT. 35Ø.)WORD=35Ø.
                        IF(WORD .LT. Ø.)WORD=Ø.
3Ø1             SAMP(I)=WORD
C
C
C
C
43Ø     WRITE(2,15Ø)BOUT1
                DO 9ØØ  J=1,256
9ØØ             ISAMP(J)=SAMP(J)+Ø.5
                WRITE(3,151)ISAMP
        NEWB=NEWB+1       !  FINAL # OF BLOCKS
345     CLOSE(UNIT=1)
        CLOSE(UNIT=2)
        CLOSE(UNIT=3)
        CLOSE(UNIT=4)
        NSAMP=NEWB*256
        OPEN(UNIT=1,TYPE='NEW',NAME='RECSAM.DAT',CARRIAGECONTROL='LIST')
        WRITE(1,888)NSAMP
888     FORMAT(I5)
        STOP
        END
```

```
C     ************************************************************
C     *                                                          *
C     *          T D H S  -  A R C  S Y S T E M                  *
C     *                R E C E I V E R                           *
C     *        PITCH IS EXTRACTED AT THE RECEIVER                *
C     *                                                          *
C     ************************************************************
C
C     QUANTIZER OUTPUT FOR FREQUENCY DIVIDED SPEECH IS RECEIVED. IT IS
C     INVERSE QUANTIZED AND PASSED THROUGH ARC RECEIVER WHICH GIVES
C     RECONSTRUCTED FREQUENCY DIVIDED SPEECH (YHAT). FREQUENCY MULTI-
C     PLICATION OPERATION IS PERFORMED ON YHAT TO GET SHAT. TO DO THIS
C     PITCH PERIOD IS NEEDED, THE VALUES OF WHICH ARE READ FROM PITCH.DAT
C     FILE. PARAMETERS AT TRANSMITTER AND THAT AT RECEIVER ARE THE SAME
C     AND ARE READ FROM PARAMETER FILE.
C
C     PROGRAM NAME:  RECVR.FTN
C     DATE         :  JUNE 30, 1981
C
C     THIS PROGRAM ASKS FOR
C                              1.  PARAMETER FILENAME: PARA.DAT
C                              2.  PITCH ESTIMATOR OPTION.
C                              3.  KBLK,ITMIN, ITMAX
C
C     AND PRODUCES OUTPUT FILE
C                              1. SHAT.DAT  (OUTPUT SPEECH)
C                              2. ZHAT.DAT  (RECONSTRUCTED COMPRSD SP)
C
      INTEGER HD(40),Q(7800),FNAME1(16),PITCH(400),P1,P2,P3,P4,SQ(16)
      INTEGER*2 FNAME2(16)
      DIMENSION YHAT(364),SHAT(364),H(400),IZHAT(256)
      DIMENSION IBUFF1(512),IBUFF2(512)
      COMMON /PRED/G,ND,RMSMIN,ALP,AINV,KQ,NSPSAM,A(12),DVHAT(12),EV,
     1 ISTAT1(40),EP,ALAD
      COMMON /RMS/RMS
      COMMON /CLIPP/CLPP
      MODN(K) = K - (K-1)/16 * 16
      MODM(K)= K - (K-1)/256 * 256
C
C
      OPEN(UNIT=8,TYPE='OLD',NAME='OPT2.DAT')
      READ(8,10)FNAME1
10    FORMAT(16A2)
C
C
      OPEN(UNIT=1, TYPE='OLD', NAME='DECO.DAT')
      OPEN(UNIT=3, TYPE='NEW', NAME='ZHAT.DAT',CARRIAGECONTROL='LIST')
      OPEN(UNIT=4, TYPE='OLD', NAME=FNAME1)
C
C
      READ(1,20)NSENT,IRATE,NSAMP,IUPPR,ILOWR,NTERMS,HD
20    FORMAT(6I5,10X,40A1)
C
      OPEN(UNIT=2,TYPE='OLD',NAME='RECSAM.DAT')
      READ(2,888)NSAMP
888   FORMAT(I5)
      NSPSAM = 2 * NSAMP
C
      TYPE *, ' TYPE THE HEADER FOR RECEIVER OUTPUT FILE:'
      ACCEPT 22,HD
22    FORMAT(40A1)
      WRITE(3,20)NSENT,IRATE,NSAMP,IUPPR,ILOWR,NTERMS,HD
C
C
```

```
      READ(1,29,END=40)(Q(J),J=1,NSAMP)
29    FORMAT(16I2)
30    FORMAT( 16I5 )
40    CONTINUE
C
C
C
      READ(8,*)CLPP
      READ(8,*)IOPT2
      READ(8,*)KBLK1,ITMIN,ITMAX
C
C
C
      CALL INSTRT
      NRMNG=MOD(NSAMP,256)
      DO 65 IBGN=1,NSAMP
      CALL ARCR(Q(IBGN),YY)
      IR1=MODM(IBGN)
      IF(YY.GT.0.0)YY=YY+0.5
      IF(YY.LT.0.0)YY=YY-0.5
      IF(YY.GT.2047.0)YY=2047.0
      IF(YY.LT.-2048.0)YY=-2048.0
      IF(IBGN.NE.1 .AND. IR1.EQ.1)WRITE(3,30)IZHAT
      IZHAT(IR1)=IFIX(YY)
      IF(IBGN.EQ.NSAMP)WRITE(3,30)(IZHAT(K),K=1,NRMNG)
65    CONTINUE
      REWIND 3
      READ(3,20)NSENT,IRATE,NSAMP,IUPPR,ILOWR,NTERMS,HD
      NSPSAM=2*NSAMP
C
C
      DO 665  I3=1,300
              Q(I3)=0
665   CONTINUE
C
C
      READ(3,30,END=68)(Q(J),J=1,NSAMP)
68    CONTINUE
      CLOSE(UNIT=3)
      OPEN(UNIT=3,TYPE='NEW',NAME='SHAT.DAT',CARRIAGECONTROL='LIST')
      WRITE(3,20)NSENT,IRATE,NSPSAM,IUPPR,ILOWR,NTERMS,HD
C
C     Q BUFFER HAS THE RECEIVED COMPRESSED SPEECH. PITCH WILL BE
C     EXTRACTED FROM IT AND WILL BE USED FOR EXPANSION.
C
C
      NUMP = 0
      IOFF = 0
      ICNTT = 0
      JJ = 1
      NADD = 0
      NUM1 = KBLK1 + ITMAX
C
C
      DO 960  I=1,NUM1
              IBUFF1(I) = Q(300+I)
960   CONTINUE
      IOFF = IOFF + NUM1
      ICNTT = IOFF
      NSINB=NUM1
998   CONTINUE
      CALL PICH(IBUFF1,NPR,ITMAX,ITMIN,KBLK1,NUMP,NSINB,IOPT2)
      PITCH(JJ) = NPR
      JJ = JJ + 1
```

```
      NADD = NADD + NPR
      NUM2 = NUM1 - NPR
C
C
      DO 980  I = 1,NUM2
              IBUFF2(I) = IBUFF1(NPR+I)
980   CONTINUE
      NSP = NUM1 - NUM2
      DO 990  I = 1, NSP
              ICNTT = ICNTT + 1
              IF( ICNTT .GT. NSAMP )  GOTO 999
              IBUFF2(NUM2+I) = Q(300+IOFF+I)
990   CONTINUE
      NSINB=NUM2+NSP
      IOFF = IOFF + NSP
      CALL PICH(IBUFF2,NPR,ITMAX,ITMIN,KBLK1,NUMP,NSINB,IOPT2)
      PITCH(JJ) = NPR
      JJ = JJ + 1
      NADD = NADD + NPR
      NUM2 = NUM1 - NPR
C
C
      DO 9100          I=1,NUM2
             IBUFF1(I) = IBUFF2(NPR+I)
9100  CONTINUE
      NSP = NUM1 - NUM2
      DO 9110   I=1,NSP
                ICNTT = ICNTT + 1
                IF(ICNTT .GT. NSAMP)GOTO 999
                IBUFF1(NUM2+I)=Q(300+IOFF+I)
9110  CONTINUE
      IOFF = IOFF + NSP
      NSINB=NUM2+NSP
      GOTO 998
999   CONTINUE
C
C
      P2 = 300
      DO 500   ICNT = 1,NUMP
             NP = PITCH(ICNT)
             P1 = P2 - NP
             P3 = P2 + NP
             P4 = P3 + NP
             P2 = P3
             DO 80    I = 0, (P4-P1-1)
                      IP11 = I + P1 + 1
                      IF( IP11 .GT. NSAMP) GOTO 600
                      YHAT(I+1) = Q(IP11)
80           CONTINUE
83           CONTINUE
             NP2 = NP + NP
             DO 100  I = 1, NP2
                      H(I) = 1 -  FLOAT(I-1)/FLOAT(NP2-1)
100          CONTINUE
             DO 120   I = 1, NP2
                      II = I
                      SHAT(II) = YHAT(II) + H(II) * (YHAT(II+NP) - YHAT(II))
                      IARG = NADD + I
                      N1 = MODN(IARG)
                      XX = SHAT(II)
                      IF(XX .GT. 0.0) XX=XX+0.5
                      IF(XX .LT. 0.0) XX=XX-0.5
                      IF(XX .GT. 2047.0)XX=2047.0
                      IF(XX .LT. -2048.0) XX=-2048.0
```

```
                        SQ(N1) = IFIX(XX)
                        IF(N1 .EQ. 16) WRITE(3,30)SQ
120             CONTINUE
        NADD = NADD + NP + NP
500     CONTINUE
C
C
600     TYPE *,' TYPE THE HEADER FOR PRINTOUT. (40 CHAR ONLY)'
        ACCEPT 775,HD
775     FORMAT(40A1)
        WRITE(6,774)HD
774     FORMAT(///,40A1,//)
        WRITE(6,776)
776     FORMAT(////6X,' SAMPLE NUMBER ',6X,' PITCH PERIOD '//)
        ISTRT = 1
        IEND  = KBLK1+ITMAX
        DO 777  I=1,NUMP
                IP=PITCH(I)
                WRITE(6,778)ISTRT,IEND,IP
                ISTRT=ISTRT+IP
                IEND =IEND+IP
777     CONTINUE
778     FORMAT(6X,I5,1X,'-',1X,I5,10X,I3)
        STOP
        END
C       ************************************************************
C       *                                                          *
C       *     I N V E R S E     Q U A N T I Z E R                  *
C       *                                                          *
C       ************************************************************
C
        SUBROUTINE INVQUA(QQ,EQ)
        INTEGER QQ
        COMMON /QUAN/T(20),OUT(20),EXPN(20),SIZE,SMIN,NQ
        COMMON /RMS/RMS
        ISIGN=1
        IF(MOD(QQ,2).EQ.0)ISIGN=-1
        J=(QQ+2)/2
        EQ=ISIGN*OUT(J)*SIZE
        SIZE=EXPN(J)*SIZE
        SIZE=AMAX1(SIZE,RMS*SMIN)
        RETURN
        END
C
C
C       ************************************
C       *                                  *
C       *  I N I T I A L I Z A T I O N     *
C       *                                  *
C       ************************************
C------ SUBROUTINE INSTRT INITIALIZES THE PARAMETERS.
C
        SUBROUTINE INSTRT
        COMMON /QUAN/T(20),OUT(20),EXPN(20),SIZE,SMIN,NQ
        COMMON /PRED/G,N,RMSMIN,ALP,AINV,KQ,NSPSAM,A(12),VHAT(12),EV
     1  ,ISTAT1(40),EP,ALAD
        COMMON /RMS/RMS
        READ(4,*)AINV,ALP,ALAD,G,N,RMSMIN,SMIN
C
        WRITE(6,2)AINV,ALP,ALAD,G,N,RMSMIN,SMIN
    2   FORMAT(/6X,'AINV=',F5.2,2X,'ALP=',F5.2,2X,'ALAD=',F5.2,2X,'G='
     1  F5.3,2X,'N=',I2,2X,'RMSMIN=',F5.1,2X,'SMIN=',F5.2/)
        READ(4,*)KQ
        WRITE(6,3)KQ
```

```
    3   FORMAT(6X,'NUMBER OF QUANT LEVELS=',I2)
        NQ=KQ/2
        NQQQ=NQ+1
        READ(4,*)(EXPN(I),I=1,NQQQ)
        WRITE(6,4)(I,EXPN(I),I=1,NQQQ)
    4   FORMAT(6X,6('EXPN(',I2,')=',F6.2,2X))
        READ(4,*)(OUT(I),I=1,NQQQ)
        WRITE(6,5)(I,OUT(I),I=1,NQQQ)
    5   FORMAT(6X,6('OUT(',I2,')=',F6.2,2X))
        SIZE=100.
        DO 118 I=1,12
        VHAT(I)=0.
  118   A(I)=0.
        RMS=RMSMIN
        A(1)=AINV
        RETURN
        END
C
C       ****************************************
C       *                                      *
C       *     A R C    R E C E I V E R         *
C       *                                      *
C       ****************************************
C
C
        SUBROUTINE ARCR(Q,VHAT1)
        INTEGER Q
        COMMON /PRED/G,N,RMSMIN,ALP,AINV,KQ,NSPSAM,A(12),VHAT(12),
     1  EV,ISTAT1(40),EP,ALAD
        COMMON /RMS/RMS
C
C------ PREDICTION.
C
C
        PRE=0.
        DO 120 I=1,N
  120   PRE=PRE+A(I)*VHAT(I)
        RMS=ALP*(RMS-RMSMIN)+(1.-ALP)*ABS(VHAT(1))+RMSMIN
        CALL INVQUA(Q,EQ)
        DO 125 I=1,N
        J=N+2-I
  125   VHAT(J)=VHAT(J-1)
        VHAT(1)=PRE+EQ
        VHAT1=VHAT(1)
C
C
C------ ADAPTATION.
C
        ERR=G*EQ/RMS**2
        A(1)=A(1)+AINV*(1./ALAD-1.)
        DO 130 I=1,N
  130   A(I)=A(I)*ALAD+ERR*VHAT(I+1)
C
C
        RETURN
        END
C
C
C       ************************************************
C       *                                              *
C       *     P I T C H     E X T R A C T I O N        *
C       *                                              *
C       ************************************************
```

```
C
C
C
      SUBROUTINE PICH( IBUF, NP, ITMAX, ITMIN,  KBLK1, NUMP,NBUFF,IOPT2)
      DIMENSION A(200), IBUFF(512), IA(200), IBUF(512)
      COMMON /CLIPP/CLPP
C
C
      NUMP = NUMP + 1
      N1 = NBUFF/3
      N2 = N1 + N1
      DO 5     I = 1,NBUFF
               IBUFF(I)=IBUF(I)
5     CONTINUE
      DO 10    I = 1, 200
              A(I) = 0.0
             IA(I) = 0
10    CONTINUE
C
C------ CHECK IF CENTER CLIPPING IS ASKED FOR.
C
      IF(IOPT2 .LE. 2) GOTO 280
C
C------ FIND OUT ABSOLUTE MAXIMUM OUT OF NBUFF SAMPLES IN THE BUFFER"IBUFF"
C
      IBIG1 = 0
      IBIG2 = 0
      DO 120   I = 1,N1
               ISPABS=ABS(IBUFF(I))
               IF(ISPABS .GT. IBIG1)IBIG1=ISPABS
120   CONTINUE
      DO 123   I = N2,NBUFF
               ISPABS=ABS(IBUFF(I))
               IF(ISPABS .GT. IBIG2)IBIG2=ISPABS
123   CONTINUE
      IB = IBIG1 - IBIG2
      IF( IB .GE. 0)IBIG=IBIG2
      IF( IB .LT. 0)IBIG=IBIG1
C
C-----  ENTER THE CLIPPING LEVEL.
C
      CL = CLPP * FLOAT(IBIG)
C
C------ CHECK IF CENTER CLIPPING WITH THREE OR TWO VALUES IS REQUIRED.
C
      IF( IOPT2 .GT. 4) GOTO 155
C
C------ CLIPP THE SPEECH WAVEFORM.
C
      CLM=-CL
      DO 140   I=1,NBUFF
               XFLT=FLOAT(IBUFF(I))
               IF((XFLT .LE. CL) .AND. (XFLT .GT. CLM))IBUFF(I)=0
140   CONTINUE
      GOTO 280
155   CONTINUE
      IF(IOPT2 .GT. 6)GOTO 360
      CLM=-CL
      DO 160   I=1,NBUFF
               XFLOT=FLOAT(IBUFF(I))
               IF((XFLOT .LE. CL) .AND. (XFLOT .GT. CLM))IBUFF(I)=0
               IF(XFLOT .GT. CL)IBUFF(I)=+1
               IF(XFLOT .LE. -CL)IBUFF(I)=-1
160   CONTINUE
```

```
      IF(IOPT2 .GT. 5)GOTO 220
C
C------ COMPUTE AUTOCORRELATION FUNCTIONS
C
      IBIGG=-1000
      DO 180  IT=ITMIN,ITMAX
              ISUM=0
              DO 170  J=1,KBLK1
              IF(((IBUFF(J+IT).LT.0).AND.(IBUFF(J).LT.0)) .OR. ((IBUFF(J+IT)
     1        .GT.0).AND.(IBUFF(J).GT.0)))ISUM=ISUM+1
              IF(((IBUFF(J+IT).LT.0).AND.(IBUFF(J).GT.0)) .OR. ((IBUFF(J+IT)
     1        .GT.0).AND.(IBUFF(J).LT.0)))ISUM=ISUM-1
170           CONTINUE
      IA(IT)=ISUM
      IF(IBIGG .LT. IA(IT)) IBIGG=IA(IT)
180   CONTINUE
      DO 190  I=ITMIN,ITMAX
              IF(IBIGG .EQ. IA(I))GOTO 200
190   CONTINUE
200   NP=I
      WRITE(5,*)NP
      RETURN
C
C-----  CALCULATE AMDF FUNCTIONS
220   CONTINUE
      ISMALL=4096
      DO 230  IT=ITMIN,ITMAX
              ISUM=0
              DO 240  J=1,KBLK1
                      IF(((IBUFF(J+IT).EQ.0).AND.(IBUFF(J).NE.0)) .OR.
     1                ((IBUFF(J+IT).NE.0).AND.(IBUFF(J).EQ.0)))ISUM=ISUM+1
                      IF(((IBUFF(J+IT).GT.0).AND.(IBUFF(J).LT.0)) .OR.
     1                ((IBUFF(J+IT).LT.0).AND.(IBUFF(J).GT.0)))ISUM=ISUM+2
240           CONTINUE
      IA(IT)=ISUM
      IF(ISMALL .GE. IA(IT))ISMALL=IA(IT)
230   CONTINUE
      DO 250  I=ITMIN,ITMAX
              IF(ISMALL .EQ. IA(I))GOTO 260
250   CONTINUE
260   NP=I
      WRITE(5,*)NP
      RETURN
280   CONTINUE
      IF((IOPT2.EQ.2).OR.(IOPT2.EQ.4))GOTO 340
      BIG=0.0
      DO 300  IT=ITMIN,ITMAX
              SUM=0.0
              DO 290  J=1,KBLK1
                      SUM=SUM+FLOAT(IBUFF(J+IT))*FLOAT(IBUFF(J))
290           CONTINUE
      A(IT)=SUM
      IF(BIG .LT. A(IT))BIG=A(IT)
300   CONTINUE
      DO 310  I=ITMIN,ITMAX
              IF(BIG.EQ.A(I))GOTO 320
310   CONTINUE
320   NP=I
      WRITE(5,*)NP
      RETURN
340   CONTINUE
      SMALL = 1.0E+09
      DO 60   IT = ITMIN, ITMAX
              SUM = 0.0
```

```
                DO 5Ø   J = 1, KBLK1
                        SUM = SUM + ABS(FLOAT(IBUFF(J+IT) - IBUFF(J)))
5Ø              CONTINUE
        A(IT) = SUM
        IF( SMALL .GE. A(IT)) SMALL = A(IT)
6Ø      CONTINUE
        DO 7Ø   I = ITMIN, ITMAX
                IF(SMALL .EQ. A(I)) GOTO 8Ø
7Ø      CONTINUE
8Ø      CONTINUE
        NP = I
        WRITE(5,*)NP
        RETURN
36Ø     CONTINUE
C       CLIPP THE SPEECH TO TWO VALUES.
        CL=Ø.Ø
        DO 38Ø  I = 1,NBUFF
                XFLOT=FLOAT(IBUFF(I))
                IF( XFLOT .LE. CL )IBUFF(I)=-1
                IF( XFLOT .GT. CL )IBUFF(I)=1
38Ø     CONTINUE
        IF(IOPT2 .GT. 7)GOTO 48Ø
        IBIGG = -1ØØØ
        DO 42Ø  IT=ITMIN,ITMAX
                ISUM=Ø
                DO 4ØØ  J=1,KBLK1
                        IF(IBUFF(J+IT) .EQ. IBUFF(J))ISUM=ISUM+1
                        IF(IBUFF(J+IT) .NE. IBUFF(J))ISUM=ISUM-1
4ØØ             CONTINUE
        IA(IT)=ISUM
        IF(IBIGG .LT. IA(IT))IBIGG=IA(IT)
42Ø     CONTINUE
        DO 44Ø  I=ITMIN,ITMAX
                IF(IBIGG .EQ. IA(I))GOTO 46Ø
44Ø     CONTINUE
46Ø     NP=I
        WRITE(5,*)NP
        RETURN
48Ø     CONTINUE
        ISMALL=4Ø96
        DO 5ØØ  IT=ITMIN,ITMAX
                ISUM=Ø
                DO 49Ø  J=1,KBLK1
                        IF(IBUFF(J+IT) .NE. IBUFF(J))ISUM=ISUM+2
49Ø             CONTINUE
        IA(IT)=ISUM
        IF(ISMALL .GE. IA(IT))ISMALL=IA(IT)
5ØØ     CONTINUE
        DO 52Ø  I=ITMIN,ITMAX
                IF(ISMALL .EQ. IA(I))GOTO 54Ø
52Ø     CONTINUE
54Ø     NP=I
        WRITE(5,*)NP
        RETURN
        END
C
C
```

REFERENCES

Anderson, J.B., and Bodie, J.B., "Tree Encoding of Speech", IEEE Trans. on Info. Theory, Vol. 11-21, July 1975.

Atal, B.S. and M.R. Schroeder, "Adaptive Predictive Coding for Speech Signals", Bell System Technical Journal, Vol. 48, No. 8, pp. 1973-1986, Oct. 1970.

Atal, B.S. and M.R. Schroeder, "Adaptive Predictive Coding for Speech Signals", Bell System Technical Journal, pp. 1973-1986, 1970.

Atal, B.S. and S.L. Hanauer, Speech Analysis and Synthesis by Linear Prediction of the Speech Wave", J. Acoust. Soc. Am., Vol. 50.2, pp. 637-655, 1971.

Atal, B.S. and M.R. Schroeder, "Predictive Coding of Speech Signals and Subjective Error Criteria", in Conf. Rec. IEEE Int. Conf. Acoustics, Speech and Signal Processing, 1978, pp. 573-576.

Barnwell, T., et al., "Tandem Interconnections of LPC and CVSD Digital Speech Coders", Final Report on DCA 100-76-C-0073, Nov. 1977.

Barnwell, T.P., "Correlation Analysis of Subjective and Objective Measures for Speech Quality", IEEE Int. Conf. on Acoustics, Speech and Signal Processing, ICASSP-80, pp. 706-709, April 1980.

Barnwell, T.P., "A Comparison of Parametrically Different Objective Speech Quality Measures Using Correlation Analysis with Subjective Qualities Results", IEEE Int. Conf. on Acoustics, Speech and Signal Processing, ICASSP-80, pp. 710-713, April 1980.

Bayless, J.W., S.J. Campanella, and A.J. Goldberg, "Voice Signals: Bit-by-Bit", IEEE Spectrum, Vol. 10, pp. 28-39, Oct. 1973.

Berger, T., "Rate Distortion Theory, A Mathematical Basis for Data Compression:, Englewood Cliffs, NJ, Prentice-Hall, 1971.

Berouti, M. and J. Makhoul, "Improved Techniques for Adaptive Predictive Coding of Speech," Proc. of ICC '78, Toronto, 1978.

Bodie, J.B., "Multi-path Tree Encoding for Analog Data Sources", Commun. Res. Lab., McMaster Univ., Hamilton, Ont., Canada, CRL Internal Rep. Series CRL-20, June 1974.

Bodie, J.B., "Multi-path Tree Encoding for Analog Data Sources", Commun. Res. Lab., McMaster Univ., Hamilton, Ont., Canada, CRL Internal Rep. Series CRL-20, June 1974.

Castellino, P., G. Modena, L. Nebbia, and C. Scagliola, "Bit Rate Reduction by Automatic Adaptation of Quantizer Step-Size in DPCM Systems", in Proc. Int. Zurch Seminar on Communications, pp. B3-1 through B6-6, 1974.

Cohn, D.L. and J.L. Melsa, "A New Configuration for Speech Digitization at 9600 Bits per Second", Proc. of 1979 Acoustic Speech and Signal Processing Conf., Washington, DC, April 1979.

Cohn, D.L. and J.L. Melsa, "The Relationship between an Adaptive Quantizer and a Variance Estimator, IEEE Trans. Inform. Theory, Vol. IT-21, pp. 669 through 671, Nov. 1975a.

Cohn, D.L. and J.L. Melsa, "The Residual Encoder -an Improved ADPCM System for Speech Digitization", IEEE Trans. Commun.. Vol. COM-23, pp. 935-941, Sept. 1975b.

Cohn, D.L. and J.L. Melsa, "Application of Sequential Adaptation to Data Rate Reduction", Proc. Eighth Asilomar Conf., Dec. 1974.

Cohn, D.L. and J.L. Melsa, "A Pitch Compensating Quantizer", Proc. 1976 IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Paper 78, April 1976.

Crochiere, R.E. and J.L. Flanagan, "Sub-Band Coding of Speech", Proc. of ICC '77, Chicago, IL, 1975.

Crochiere, R.E., S.A. Webber and J.L. Flanagan, "Digital Coding of Speech in Sub-bands", Bell Syst. Tech. J., 55, 1069-1085, Oct. 1976.

Crochiere, R.E., "On the Design of Sub-band Coders for Low Bit Rate Speech Communications", Bell Syst. Tech. J., May 1977.

Crochiere, R.E. and M.R. Sambur, "A Variable Band Coding Scheme for Speech Encoding at 4.8 kb/s", to be published B.S.T.J., see also Proc. IEEE Int. Conf. on Acoust., Speech, and Signal Proc., 1977.

Crochiere, R.E., "An analysis of 16 kb/s Sub-band Coder Performance: Dynamic Range, Tandem Connections and Channel Errors", Bell Syst. Tech. J., Vol. 57, pp. 2927-2952, Oct. 1978.

Cummiskey, P., N.S. Jayant and J.L. Flanagan, "Adaptive Quantization in Differential PCM Coding of Speech", Bell Syst. Tech. J., pp. 1105-1118, Sept. 1973.

Cummiskey, P., N.S. Jayant and J.L. Flanagan, "Adaptive Quantization in Differential PCM Coding of Speech", Conf. Rec., 1973 IEEE Int. Conf. Communications, pp. 46/17-46/22.

Dunn, J.G., "Experimental 9600-bits/s Voice Digitizer Employing Adaptive Prediction", IEEE Trans. Commun. Technol., Vol. COM-19, pp. 1021-1032, Dec. 1971.

Flanagan, J.L., Speech Analysis, Synthesis and Perception, 2nd Ed., Springer-Verlag, New York, 1972.

Flanagan, J.L., et al., "Speech Coding", IEEE Trans. Commun., Vol. COM-27, pp. 710-737, April 1979.

Flanagan, J.L. and R.M. Golden, "Phase Vocoder", Bell Syst. Tech. J., Vol. 45, pp. 1493-1509, Nov. 1966.

Frei, A.H., H.R. Schindler, P. Vettiger and E. Von Felton, "Adaptive Predictive Speech Coding Based on Pitch Controlled Interruption/reiteration Techniques", in Proc. IEEE Int. Conf. Communications, Seattle, WA, June 1973, pp. 46/12-46/16.

Gallagher, R.G., "Tree Encoding for Sources with a Distortion Measure", IEEE Trans. Inform. Theory, Vol. IT-20, pp. 65-76, Jan. 1974.

Gallagher, R.G., Information Theory and Reliable Communication, New York: John Wiley, 1968.

Gibson, J.D., S.K. Jones and J.L. Melsa, "Sequentially Adaptive Prediction and Coding of Speech Signals", IEEE Trans. on Commun., Vol. COM-22, pp. 1789-1796, Nov. 1974.

Gibson, J.D. and J.L. Melsa, "Unified Development of Algorithms Used for Linear PRedictive Coding of Speech Signals", J. of Computer and Elec. Engr., Vol. 3, pp. 75-91, 1976.

Gibson, J.D., J.L. Melsa and S.K. Jones, "Digital Speech Analysis Using Sequential Estimation Techniques", IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. ASSP-23, No. 4, Aug. 1975.

Gibson, J.D., "Adaptive Prediction in Speech Differential Encoding Systems", Proc. of IEEE, Vol. 68, No. 4, April 1980.

Gold, B. and L.R. Rabiner, "Parallel Processing Techniques for Estimating Pitch Periods of Speech in the Time Doamin", J. Acoust. Soc. Am., Vol. 46, No. 2, Pt. 2, pp. 442-448, Aug. 1969.

Goldberg, A.J., et al., "Kalman Predictive Encoder", Final Report on DCA 100-74-C-0058, July 1975.

Goldberg, A.J., R.L. Freudberg, and R.S. Cheung, "High Quality 16 kb/s Voice Transmission", IEEE Int. Conf. on Acoustics, Speech and Signal Processing, pp. 244-246, Spring 1976.

Goldberg, A.J., L.E. Bergeron, S.Y. Kwon and M. Miller, "A Robust Adaptive Transform Coder for 9.6 kb/s Speech Transmission", IEEE Int. Conf. on Acoustics, Speech and Signal Processing, pp. 344-347, April 1980.

Goodman, D.J. and A. Gersho, "Theory of an Adaptive Quantizer", IEEE Trans. Commun., Vol. COM-22, pp. 1037-1046, Aug. 1974.

Goodman, D.J. and R.M. Wilkinson, "A Robust Adaptive Quantizer", IEEE Trans. Commun., Vol. COM-23, pp. 1362-1365, Nov. 1975.

Greefkes, J.A. and F. de Jager, "Continuous Delta Modulation", Phillips Res. Rept., No. 23, pp. 233-246, 1968.

Hamming, R.W., Coding and Information Theory, Prentice-Hall, 1980.

Itakura, F. and S. Salto, "Analysis Synthesis Telephony Based on the Macimum Likelihood Method", Reports of the 6th International Congress on Acoustics, Tokyo, Japan, Edited by Y. Kohasi, pp. C17-C20, Paper C-5-5, Aug. 21-28, 1968.

Jayant, N.D., "Adaptive Quantization with a One-Word Memory", Bell System Technical Journal, Vol. 52, No. 7, pp. 1119-1144, Sept. 1973.

Jayant, N.D., "Digital Coding of Speech Waveform, PCM, DPCM, and DM Quantizers", IEEE Proc., Vol. 62, No. 5, pp. 611-632.

Jelinek, F. and J.B. Anderson, "Instrumentable Tree Encoding of Information Sources", IEEE Trans. Inform. Theory, Vol. IT-17, pp. 118-119, Jan. 1971.

Licklider, J.C.R. and I. Pollack, "Effects of Differentiation, Integration and Infinite Peak Clipping upon Intelligibility of Speech", J. Acoust. Soc. Am., Vol. 20, pp. 42-50, Jan. 1948.

McDonald, R.A., "Signal-to-Noise and Idle Channel Performance of Differential Pulse Code Modulation Systems -Particular Applications to Voice Signals", Bell Syst. Tech. J., Vol. 45, pp. 1123-1151, Sept. 1966.

Malah, D., "Time-domain Algorithms for Harmonic Bandwidth Reduction and Time-Scaling of Speech Signals", IEEE Trans. Acoust., Speech and Signal Proc., Vol. ASSP-27, No. 2, pp. 121-133, April 1979.

Malah, D., "Combined Time-Domain Harmonic Compression and CVSD for 7.2 kbit/s Transmission of Speech Signals", 1980 ICASSP, Denver, CO, pp. 504-507, April 1980.

Malah, D., R.E. Crochiere and R.V. Cox, "Performance of Transform and Sub-band Coding Systems Combined with Harmonic Scaling of Speech", to be published in IEEE ASSP Trans., 1981.

Makhoul, J. and J.J. Wolf, "Linear Prediction and the Spectral Analysis of Speech", NTIS No. AD-749066, Report No. 2304, Bolt Beranek and Newman Inc., Cambridge, MA, Aug. 1972.

Makhoul, J., "Linear Prediction: A Tutorial Review", IEEE Proceedings, Vol. 63, No. 4, April 1975.

Makhoul J. and M. Berouti, "Adaptive Noise Spectral Shaping and Entropy Coding in Predictive Coding of Speech", IEEE Trans. Acoust., Speech and Signal Proc., Vo. ASSP-27, pp. 63-73, Feb. 1979.

Goodman, D.J. and R.M. Wilkinson, "A Robust Adaptive Quantizer", IEEE Trans. Commun., Vol. COM-23, pp. 1362-1365, Nov. 1975.

Greefkes, J.A. and F. de Jager, "Continuous Delta Modulation", Phillips Res. Rept., No. 23, pp. 233-246, 1968.

Hamming, R.W., Coding and Information Theory, Prentice-Hall, 1980.

Itakura, F. and S. Salto, "Analysis Synthesis Telephony Based on the Macimum Likelihood Method", Reports of the 6th International Congress on Acoustics, Tokyo, Japan, Edited by Y. Kohasi, pp. C17-C20, Paper C-5-5, Aug. 21-28, 1968.

Jayant, N.D., "Adaptive Quantization with a One-Word Memory", Bell System Technical Journal, Vol. 52, No. 7, pp. 1119-1144, Sept. 1973.

Jayant, N.D., "Digital Coding of Speech Waveform, PCM, DPCM, and DM Quantizers", IEEE Proc., Vol. 62, No. 5, pp. 611-632.

Jelinek, F. and J.B. Anderson, "Instrumentable Tree Encoding of Information Sources", IEEE Trans. Inform. Theory, Vol. IT-17, pp. 118-119, Jan. 1971.

Licklider, J.C.R. and I. Pollack, "Effects of Differentiation, Integration and Infinite Peak Clipping upon Intelligibility of Speech", J. Acoust. Soc. Am., Vol. 20, pp. 42-50, Jan. 1948.

McDonald, R.A., "Signal-to-Noise and Idle Channel Performance of Differential Pulse Code Modulation Systems -Particular Applications to Voice Signals", Bell Syst. Tech. J., Vol. 45, pp. 1123-1151, Sept. 1966.

Malah, D., "Time-domain Algorithms for Harmonic Bandwidth Reduction and Time-Scaling of Speech Signals", IEEE Trans. Acoust., Speech and Signal Proc., Vol. ASSP-27, No. 2, pp. 121-133, April 1979.

Malah, D., "Combined Time-Domain Harmonic Compression and CVSD for 7.2 kbit/s Transmission of Speech Signals", 1980 ICASSP, Denver, CO, pp. 504-507, April 1980.

Malah, D., R.E. Crochiere and R.V. Cox, "Performance of Transform and Sub-band Coding Systems Combined with Harmonic Scaling of Speech", IEEE ASSP Trans., vol. ASSP-29, no. 2, pp. 273-283, April 1981.

Makhoul, J. and J.J. Wolf, "Linear Prediction and the Spectral Analysis of Speech", NTIS No. AD-749066, Report No. 2304, Bolt Beranek and Newman Inc., Cambridge, MA, Aug. 1972.

Makhoul, J., "Linear Prediction: A Tutorial Review", IEEE Proceedings, Vol. 63, No. 4, April 1975.

Makhoul J. and M. Berouti, "Adaptive Noise Spectral Shaping and Entropy Coding in Predictive Coding of Speech", IEEE Trans. Acoust., Speech and Signal Proc., Vo. ASSP-27, pp. 63-73, Feb. 1979.

O'Neal, J.B., Jr. and R.W. Stroh, "Differential PCM for Speech and Data Signals", IEEE Trans. Commun. Technol., Vol. COM-20, pp. 900-912, Oct. 1972.

O'Neal, J.B. and R.W. Stroh, "Differential PCM for Speech and Data Signals", IEEE Trans. Commun., Vol. COM-20, pp. 900-912, May 1972.

Qureshi, S.U.H. and G.D. Forney, Jr., "A 9.6/16 kb/s Speech Digitizer", Conf. Record 1975 International Conf. on Communications, San Francisco, CA, 1975.

Rabiner, L.R. and R.W. Schafer, "Design of Digital Filter Banks for Speech Analysis", Bell Syst. Tech. J., Vol. 50, pp. 3097-3115, Dec. 1971.

Rabiner, L.R., R.W. Schafer and J.J. Dubnowski", Real-Time Digital Hardware Pitch Detector", IEEE Trans. Acoust., Speech and Signal Proc., Vol. ASSP-24, No. 1, pp. 2-8, Feb. 1976.

Rabiner, L.R. and R.W. Schafer, Digital Processing of Speech Signals, Prentice-Hall, Inc., New Jersey, 1978.

Rabiner, L.R., "On the Use of Autocorrelation Analysis for Pitch Detection", IEEE Trans. Acoust., Speech and Signal Proc., Vol. ASSP-24, No. 1, pp. 2-8, Feb. 1977.

Ross, M., et al., "Average Magnitude Difference Function Pitch Extractor", to be published in the IEEE Trans. on Acoustics, Speech and Signal Proc., 1974.

Sambur, M.R., "High Quality 9.6 KBPS Algorithm that Satisfies the Embedded Bit Concept", Proc. of ICC '78, Toronto, 1978.

Sambur, M.R., "Recent Advances in LPC Speech Vocoding", Proc. of ICC '77, Chicago, IL, 1975.

Schafer, R.W. and L.R. Rabiner, "Design and Simulation of a Speech Analysis-Synthesis System Based on Short-Time Fourier Analysis", IEEE Trans. Audio Electroacoust., Vol. AU-21, pp. 165-174, June 1973.

Schroeder, M.R., J.L. Flanagan and E.A. Lundry, "Bandwidth Compression of Speech by Analytic-Signal Rooting", Proc. IEEE, Vol. 55, pp. 396-401, March 1967.

Steele, R., Delta Modulation Systems, Halsted Press, London, 1975.

Steiglitz, K., "On the Simultaneous Estimation of Poles and Zeros in Speech Analysis", IEEE Trans. in Acoustics, Speech, and Signal Processing, Vol. ASSP-25, No. 3, June 1977.

Stroh, R.W., "Differential PCM with Adaptive Quantization for Voice Communications", Proc. Int. Conf. Communications, Paper 13C, pp. 1-5, June 1971.

Sondhi, M.M., "New Methods of Pitch Extraction", IEEE Trans. Audio and Electroacoust., Vol. AU-16, No. 2, pp. 262-266, June 1968.

Tomcik, J.D., "An Application of State and Parameter Estimation Techniques for the CVSD to the LPC Interconnection Problem", Ph.D. Dissertation, Univ. of Notre Dame, May 1979.

Tomcik, J.D. and J.L. Melsa, "Identification of Speech and Speech Parameters in a Noisy Environment", Proc. of 1977 JACC, San Francisco, CA, June 1977a.

Tomcik, J.D. and J.L. Melsa, "Least Squares Estimation of Predictor Coefficients from Noisy Observations", Proc. of 1977 IEEE CDC, New Orleans, LA, Dec. 1977b.

Uddenfeldt, J., "A Performance Bound for Tree Search Coding of Speech with Minimum-Phase Codes", Proc. of ICC '78, Toronto, 1978.

Un, C.K. and D.T. Magill, "The Residual-Excited Linear Prediction Vocoder with Transmission Rate Below 9.6 bits/s", IEEE Trans. Comm., Vol. COM-23, No. 12, pp. 1460-1474, 1975.

Virupaksha, K. and J.B. O'Neal, Jr., "Entropy-coded Adaptive Differential Pulse-code Modulation (DPCM) for Speech", IEEE Trans. Comm., Vol. COM-22, pp. 777-787, June 1974.

Viswanathan, R., W. Russell, A. Higgins and J. Makhoul, "Baseband LPC Coders for Speech Transmission over 9.6 kb/s Noisy Channels", IEEE Int. Conf. on Acoustics, Speech and Signal Proc., pp. 348-351, April 1980.

Wilson, S.G. and S. Husain, "Tree Encoding of Speech at 8000 bps with a Frequency weighted Fidelity Criterion", Preprint of text submitted to IEEE Trans. Comm.

Zelinski, R. and P. Noll, "Adaptive Transform Coding of Speech Signals", IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. ASSP-25, pp. 299-309, Aug. 1977.

END

DATE
FILMED
11-81

DTIC